

DESIGN ENVIRONMENT USER GUIDE

Vector Fields Limited
24 Bankside
Kidlington
Oxford OX5 1JE
England

Copyright © 1999-2002 by Vector Fields Limited, England

This document was prepared using Adobe® FrameMaker®.

UNIX is a trademark of X/Open Company Ltd.

HPGL is a trademark of Hewlett-Packard Incorporated

PostScript is a trademark of Adobe Systems Incorporated

Windows 95/98/NT is a trademark of Microsoft Corporation

X Window System is a trademark of Massachusetts Institute of Technology

I-DEAS is a trademark of Structural Dynamics Research Corporation

All other brand or product names are trademarks or registered trademarks
of their respective companies or organisations.

CONTENTS

Chapter 1

Structure of the User Guide

Road Map	1-1
----------------	-----

Chapter 2

Introduction to the Design Environment

Program Philosophy	2-1
The Design Process	2-3
Constraining the Model	2-8

Chapter 3

Induction Sensor User System

The User System	3-1
-----------------------	-----

Chapter 4

Induction Sensor Configuration

Induction Sensor	4-1
The Core	4-4
The Coil	4-13
The Target	4-19
The Background	4-24
Boundary Conditions	4-27
Checking the model	4-28

Setting analysis options	4-29
Setting the Post Processing	4-30
Preparing PARTS for the User System	4-32
Model Testing	4-45

Chapter 5

Switched Reluctance Motor User System

The User System	5-1
-----------------------	-----

Chapter 6

Switched Reluctance Motor Configuration

Switched Reluctance Motor (SRM)	6-1
The rotor and airgap	6-3
The stator and windings	6-28
Motor	6-52
Model testing	6-59

Chapter 1

Structure of the User Guide

Road Map

The OPERA-2d Design Environment User Guide is structured into the following chapters.

Introduction

This chapter introduces the Design Environment, the program philosophy and the key design concepts. The use of parameterised models is presented by looking at the typical design process. The types and use of constraints are finally shown, along with some tips to get the most out of the Design Environment.

Induction Sensor User System

As an example of the ease of use and power of the *Design Environment*, the *User System* is demonstrated. Several model variations are generated, analysed and post processed using the facilities provided within the *Design Environment User System*. This example uses the supplied *Design Environment Module*, *sensor.dem*, for modelling of the induction sensor.

Induction Sensor Configuration

This chapter demonstrates the Configuration System and describes the steps necessary to generate a fully parameterised model. An induction sensor is used as the parametric model for this example, with the Steady State (AC) module being required for analysis.

Switched Reluctance Motor User System

A second example of the *Design Environment Configuration System* is demonstrated. A switched reluctance motor is used as the parametric model for this example, with the Static analysis (ST) module being used for analysis. Several model variations are generated, analysed and post processed using the facilities provided within the *Design Environment User System*. This example uses the supplied *Design Environment Module*, *srm.dem*, for modelling of the motor.

Switched Reluctance Motor Configuration

This chapter describes the steps necessary to generate the fully parameterised model used in the previous chapter.

Chapter 2

Introduction to the Design Environment

Program Philosophy

The Design Environment provides facilities for creating parameterized models and for adjusting these models using a graphical user interface (GUI).

The key to a parameterised model is the definition of numerical data through variables and expressions. All material data, boundary conditions and the finite element mesh definitions can be set as parameters. The geometry of any device is defined through the use of constraints. There are 9 types of constraints and these define geometric relationships linking the various points of the device. This method of defining a device's geometry is powerful and flexible and allows for complete control.

System Overview

The *Design Environment* offers a two stage approach, whereby an 'expert' sets up the environment, and a designer, perhaps less familiar with the software, can use this environment to carry out complex finite element analysis.

The *Configuration System* is the software used by the 'expert'. This system allows for the definition of the variables and expressions on which the model is based. Geometric constraints, material data and post processing are also defined so as to generate a Design Environment Module. The configuring of the post processing is best undertaken by those who have had some previous experience of using the standard OPERA-2d post processor.

The *Design Environment Module* is a data file containing the complete specification of the design of a particular class of device. It includes adjustable data, post

processing details and the GUI data necessary to define the environment of the User System.

The *User System* is a program for the designer. It allows the generation of device variations through a simple and intuitive interface. Having adjusted a design by altering a set of parameters, the User System outputs a standard pre processing file and post processing data file. This data can then be solved using the finite element analysis modules available and the data can subsequently be post processed automatically.

Location of files

The sample Design Environment Module (DEM) files for sections 2 and 4 can be found in the *work/examples/2d* subdirectory of the installation directory.

The Design Process

Using the Design Environment

This section aims to introduce the main features and structures that are used in the Design Environment and to give an understanding of the process required to generate a parameterised model. The following sections will look in more detail at the use of variables and expressions, the use of constraints to define a geometry and how these all work together to generate a powerful and flexible parametric model.

The process of generating a parametric model can be divided into several consecutive sections. Firstly, the variables used to define the model need to be defined. Secondly the geometry needs to be generated and constrained. Thirdly, the material properties must be defined. Finally, the User System that enables straightforward use of the parametric model needs to be set up. This User System includes the provision of automatic post processing and the definition of parts of the GUI.

It is important to note that the process of generating such a model is necessarily more involved than using the standard pre processor, but that the subsequent saving in time and effort through the generation of design variations will always be appreciated.

Variables and expressions

The *Design Environment* allows the user to create variables using the **VARIABLE** command. These variables can be used individually, or as part of larger expressions, to define material or geometric characteristics. The variables are the foundation and building blocks of a parametric model. In many instances, it is important to plan a parametric model by identifying the primary variables, which take numeric values, and secondary variables, which take expressions involving the primary variables as their values.

Of course, there are some design areas that are fixed and do not require the use of variables. These design areas can have direct numeric data assigned and will subsequently not be as straightforward to modify, since they cannot be changed by simply altering a variable value. The user has complete control over the areas and the extent to which a model exhibits parameterisation. If a 'one-off' model is required, it may well be more sensible to use the standard OPERA-2d pre processor as the means of model preparation.

All expressions are entered in a standard format and use most of the FORTRAN 77 mathematical functions.

Polygons, polygon points and reference points

All models generated in the Design Environment are composed of polygons, which differentiate between regions exhibiting varying material properties. The Design Environment differs from the standard pre processor in asking the user to generate complete polygons as a starting point. These polygons can have any number of facets (up to 99), and are by default regularly shaped and defined to be air.

These polygons will need to have their geometries constrained by the placement of their nodes. The material properties (permeability, conductivity, etc.) will also be defined using parametric expressions.

All polygons have their nodes uniquely named. The polygon nodes take their name from the polygon number and their position within the polygon. For example, point **2.04** is the fourth point in polygon 2. Polygons can have up to 99 nodes. These node names are used as part of the constraint expressions, which are explained in the next section.

Once polygons have been constrained, they can be copied or replicated. Replications are defined by the **MATERIAL** command and can be parametric themselves. For example, the number of rotational replications of a polygon could be defined by an expression. A replicated polygon will almost always exhibit the identical geometric shape and identical material data as the original. The one exception is when the **INDEX** variable is used as part of a material expression.

A polygon can be copied so as to generate an independent polygon which exhibits alternative material or geometric properties. A copied polygon will be constrained in a different way from the original polygon. Initially, the constraints relate the original polygon to the copied polygon, so that changes in the original polygon will be seen in the copied polygon. The default constraints that are generated as part of the copying process can be deleted and new constraints applied to achieve greater geometric independence. All material definitions, including parametric expressions, will be transferred to the copied polygon. These expressions can be altered and modified to achieve independent material definitions. Similarly, all replication data will be transferred to the copied polygon and these can be subsequently modified.

Occasionally it is useful to be able to set up a constraint from a point which is not an essential part of the model. In this case, the point may be defined as a reference point. As its name suggests, a reference point only exists to provide a point of ref-

erence. It does not contribute any material or geometric information to the model, except when it is used as a mesh control point (see the **GRAVITY** command in the reference manual). Up to 1000 reference points may be defined, and their name can be seen as a decimal number between 0 and 1. For example, a reference point could have the name **0.023**.

Constraints

A constraint is a specific relationship between coordinates of one or more polygon nodes. Each constraint relationship can be defined using a parametric expression.

There are 9 different constraint types that can be used to relate nodes to one another. All the constraints take polygon node or reference point names and parametric expressions as their input data.

- Fixed Point (**POINT** command)

This constraint takes a single point name and places it at a (x,y) coordinate location, where the x and y values are defined as parametric expressions. This constraint resembles the definition of points using the standard pre processor. All models need at least one point constraint, since this type of constraint will place the model at some unique cartesian point.

- Vector (**VECTOR** command)

This constraint takes two point names and places them at a vector (x,y) apart from each other. The values of the x and y vector components are set by parametric expressions. The order of the point names is important, as the defined vector starts at point 1 and finishes at point 2.

- Vector Difference (**DVECTOR** command)

This constraint takes 4 point names. The x-vector components between points 1-2, and 3-4 are defined to have a difference set by a parametric expression. A second parametric expression is required to define the difference in the y-vector components. Generally, the constraint is used with both expressions being zero, i.e. to generate parallel sides of the same length.

- Length (**LENGTH** command)

This constraint takes 2 point names and sets a length between them. This length takes its value from a parametric expression. This constraint is in itself insufficient to uniquely define a point, since some angular data is required. Hence this constraint is frequently used in conjunction with the angle constraints.

- Length Difference (**DLENGTH** command)

This constraint takes 4 point names. The lengths between points 1-2 and 3-4 are compared and the difference set to the parametric expression. This con-

straint is in itself insufficient to uniquely define a point. The constraint is generally used with the expression set to zero so that the two lengths are identical.

- Angle (**ANGLE** command)

This constraint takes 2 point names and fixes their positions such that they form an angle with respect to the global x-axis. The angle is set by a parametric expression. This constraint is frequently used in conjunction with the length constraint, since by itself it cannot uniquely position a point.

- Angle Difference (**DANGLE** command)

This constraint takes 4 point names. The absolute angles between points 1-2 and 3-4 are compared and the difference set to the parametric expression. This constraint is frequently used with the parametric expression set to zero so as to ensure that the 2 angles defined by 4 points are identical, i.e. parallel lines can be generated.

- Internal Angle (**IANGLE** command)

This constraint takes 3 point names and defines the angle generated by the intersection of the 2 lines between points 1-2 and 2-3. The second point is considered to be the intersection point of the two lines. The internal angle is defined by a parametric expression. The order of the points is important, with the point 2 always being considered the point at which the internal angle is measured.

- Internal Angle Difference (**DIANGLE** command)

This constraint takes 6 point names. The internal angles between points 1-2-3 and 4-5-6 are compared and the difference set to the parametric expression. This constraint is frequently used to ensure that two angular aspects of a model are identical.

NB. A parametric model can be set up through several alternate constraint schemes, which although different, achieve the same net effect. Some schemes may be more preferable than others (see [“Constraining the Model” on page 2-8](#)).

Material and mesh properties

All material data (e.g., permeability, conductivity, current density, etc.) can be defined using parametric expressions within the **MATERIAL** command.

One important material parameter (**SHAPE**) sets whether the polygon is a background polygon. The Design Environment allows only one background polygon to exist at any one time.

The finite element mesh is generally defined by a parameter (**SUBDIVISION**) that sets the maximum element size within a polygon. This parameter is defined

for the whole polygon. Polygons may have different mesh densities defined. The Design Environment ensures that the finite element mesh is continuous across neighbouring polygons.

The mesh density may be altered at a particular point by using the **GRAVITY** command. This allows a local variation in mesh density to be defined at particular points of interest.

Post Processing (**POST** command)

Post processing can be set up within the Design Environment to characterise specific design areas. The post processing uses the same commands and features that are present in the standard OPERA-2d post processor. Helpful differences include the use of point names instead of the more usual (x,y) coordinate definitions for commands such as **INTLINE**. This allows the post processing to follow any changes in the model geometry. The post processing is carried out through the generation of a configuration-specific command file (*.comi) which is subsequently used within the standard OPERA-2d post processor.

Constraining the Model

Handling Constraints

Constraining the model correctly will be one of the more important parts of using the Configuration System to build a parameterised model. There are a few hazards to be aware of when constraining a model. In general, if these hazards are avoided, constraining the model will be relatively straightforward. Before starting to apply any constraints to a model it is necessary to have planned a scheme for dimensioning the model in advance.

Wherever possible, use **POINT** and **VECTOR** type constraints, since these can rarely, if ever, go wrong. These constraints are the simplest, fastest to solve and always provide a unique constraint scheme. Whenever a **LENGTH** constraint is employed, try to constrain the same point with a **ANGLE** constraint. If these two straightforward rules are obeyed, there should be little worry in encountering some of the hazards described below.

Constraint Schemes

When constraining, it is necessary to create a suitable set of constraints, that will allow model variation in agreement with your dimensioning scheme. To achieve this, the set of constraints must allow *exact* calculation of all unknowns within the model. The unknowns in the model are the (x,y) coordinates of each node. Therefore, for a set of **n** nodes, there are **2n** unknowns.

Fully constrained model

To allow calculation of these **2n** unknowns will require a set of **2n** constraint equations, at which point the model is said to be *fully constrained*.

Under constrained model

When setting up the constraints, parts of the model will not be fully constrained. The model is said to be *under-constrained*. When under-constrained, the geometry is not unique and further constraints must be added.

Over- constrained model

More than **2n** constraints leads to *over-constraining* where one or more of the constraints are unused as all unknowns have already been fixed. One or more constraints must be removed to make it fully constrained.

Over-constraining part of the model is also possible. In this case, there may be fewer than $2n$ constraints in total, but they are not evenly spread through the model, so some sections of the model are over-constrained, other sections are under-constrained.

Redundant or inconsistent constraining

This can occur when a constraint is already implicitly implied by other constraints. An example is where the three internal angles of a triangle are specified. Only 2 angles need be specified and the 3rd is known. Giving all three produces a redundancy if the three angles sum to 180° or an inconsistency otherwise.

Non-unique solution

Non unique solutions arise from the solution of non-linear equations. A simple case is where a point is specified by distances from 2 points. In this case there are 2 possible solutions available as shown in Figure 2.1 and there is no way to distinguish the correct solution. The solution found will depend upon the proximity of the previous points to one of these solutions.

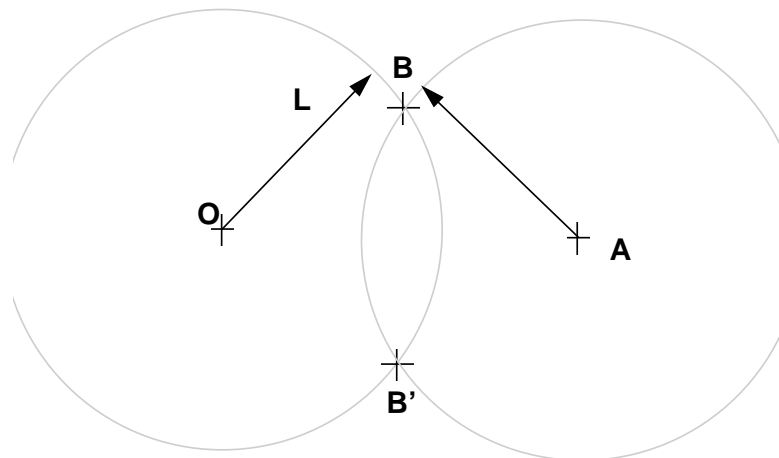


Figure 2.1 A non-unique solution

A special case of non-unique solutions exists for the angle constraints. This occurs when a solution is found in which an angle is indeterminate because of a zero length, i.e. what is the angle between coincident points?

Occasional Inconsistency

This occurs when a change in a constraint expression means that no solution to the set of constraints exists. Certain constraint combinations are never susceptible

to this form of failure and should be used where possible. To help the user, limits can be set (see the **LIMIT** command) that restrict the value of expressions.

Figure 2.2 shows a potential inconsistency. Points **O** and **A** are fixed. Point **B** is constrained to lie at length **L** from both points **O** and **A**. If the length of the constraint from point **O** is reduced to **L'**, then it can be seen that no solution is now possible from this constraint set.

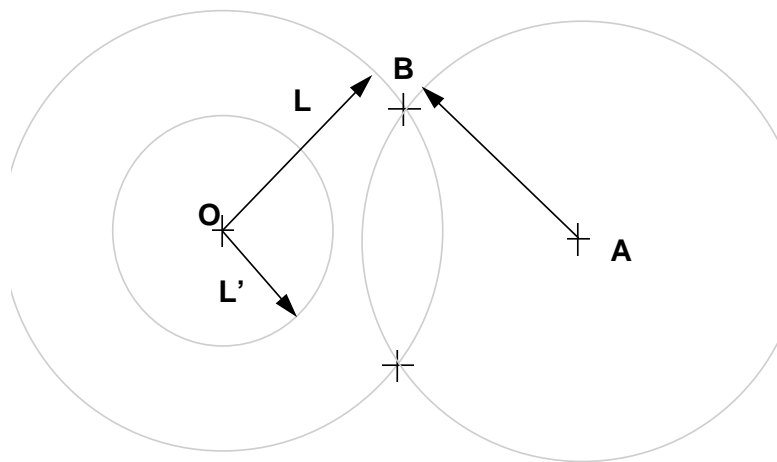


Figure 2.2 A potentially inconsistent solution

The Configuration System provides an interactive means of entering the constraint scheme. It is able to tell you where over and under-constraining is occurring. As constraints are added into the model, new solutions will be found. Any redundant or inconsistent constraints found, will be highlighted. Listing the polygons will show which sections of the model still need to be constrained.

Unfortunately, there is no way of finding multiple solutions or of spotting constraints that might become inconsistent when the expression is changed. The user must be careful to try to avoid these cases, or to add limiting equations where possible.

Constraint definitions

There are nine types of constraint available to the user of the Configuration System. These should be sufficient to generate a wide range of parameterized model.

All follow a very standard input form, requiring a type of constraint, an ordered list of points affected by the constraint, and an expression (some require 2 expressions as they generate 2 equations).

The first 3 constraint types all require 2 expressions (since they are based on a cartesian geometry system, and thus require expressions for x and y values). In each case the values entered for xp and yp are expressions. In these the order of points used is *very* important as the vector **p1** to **p2** is different from vector **p2** to **p1**.

- Point

POINT P1 XP YP

This is the simplest constraint form. It effectively sets the X and Y coordinates to the value given by the expressions XP and YP. Two equations are generated from this constraint:

$$\begin{aligned} X_1 - XP &= 0 \\ Y_1 - YP &= 0 \end{aligned} \quad (2.1)$$

This is a very direct constraint to use and should be used when possible. At least one POINT constraint *must* be used in a model to fix its position in free space.

- Vector

VECTOR P1 P2 XP YP

This is also a very straightforward constraint and should be used where possible. Two equations are generated to show that the vector from **p1** to **p2** is (xp, yp), i.e.

$$\begin{aligned} X_2 - X_1 - XP &= 0 \\ Y_2 - Y_1 - YP &= 0 \end{aligned} \quad (2.2)$$

- Difference in vectors

DVECTOR P1 P2 P3 P4 XP YP

This command sets the difference between the vector **p1** to **p2** and the vector **p3** to **p4** to the vector (xp,yp).

$$\begin{aligned} (X_2 - X_1) - (X_4 - X_3) - XP &= 0 \\ (Y_2 - Y_1) - (Y_4 - Y_3) - YP &= 0 \end{aligned} \quad (2.3)$$

This is a useful constraint for generating parallel lines of the same length by setting xp = yp = 0.

The next 2 types of constraint are length based constraints. Each takes a single expression and generates a single equation in the constraint scheme.

- Length

LENGTH P1 P2 D

This sets the absolute difference between points **p1** and **p2** to the value **d**. A single equation is generated by the constraint

$$(X_1 - X_2)^2 + (Y_1 - Y_2)^2 - d = 0 \quad (2.4)$$

This quadratic equation can lead to multiple solutions. Care should be used to try to ensure that only one solution is possible. For example, use of **ANGLE** constraints that include points **p1** and **p2** creates a single solution.

- Difference in Length

DLENGTH P1 P2 P3 P4 D

This sets the absolute difference between lengths **p1** to **p2**, and **p3** to **p4** to the value **d**. A single equation is generated by the constraint

$$\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2} - \sqrt{(X_3 - X_4)^2 + (Y_3 - Y_4)^2} - d = 0 \quad (2.5)$$

Care must be taken with the order of points being given. It generates a single equation, which can lead to multiple solutions.

It is useful for specifying circular type objects, e.g. where a radial distance is the same ($d=0$), or where there is a radial gap (e.g. an air gap in a motor).

The next 4 types of constraint are angle based constraints. Each takes a single expression and generates a single equation in the constraint scheme. All angles should be specified in DEGREES.

In each case, care should be taken to try to avoid non-unique solutions caused by lengths collapsing to 0. Although a warning is given to highlight this problem, it may occasionally prevent a true solution being found.

- Angle

ANGLE P1 P2 θ

This constrains the angle from point **p1** to point **p2** to be an absolute angle θ relative to the positive X axis.

$$\text{atan}\left(\frac{Y_2 - Y_1}{X_2 - X_1}\right) = \theta \quad (2.6)$$

This constraint type is useful for radial type geometries, e.g. rotors, where by changing the expression for θ , the rotor will rotate.

- Difference in angles

DANGLE P1 P2 P3 P4 θ

This constrains the difference between angles of the lines from point **p1** to point **p2**, and from **p3** to **p4** to be an angle θ .

$$\text{atan}\left(\frac{Y_2 - Y_1}{X_2 - X_1}\right) - \text{atan}\left(\frac{Y_4 - Y_3}{X_4 - X_3}\right) = \theta \quad (2.7)$$

This constraint creates a relative angle between parts of the model. Setting $\theta=0$ creates parallel lines (in the same direction!). This can be used when one section of the model is dependent on the angle of another part, i.e. rotation of coils in the armature.

- Internal angle

IANGLE P1 P2 P3 θ

This is a special variation of the **DANGLE** constraint, and is in fact equivalent to

DANGLE P2 P1 P3 P1 θ

It sets the internal angle generated from going from points **p1** to **p2** to **p3** to be θ . The most common use is for right angles, where $\theta=90$.

- Difference in internal angles

DIANGLE P1 P2 P3 P4 P5 P6 θ

This sets the difference in internal angles between angle **p1-p3** and **p4-p6** to be θ . This is useful for the same internal angle where $\theta=0$.

Chapter 3

Induction Sensor User System

The User System

This chapter will show the use of a pre-defined model of an induction sensor, with a typical design shown in Figure 3.1. A parameterised model of this device has been generated using the Design Environment and is stored in a file called *sensor.dem*. This file can also be found in the OPERA-2d installation sub-folder *Examples/2D*. The details of building this model is given in “[Induction Sensor Configuration](#)” on page 4-1

The device is axisymmetric in nature. There are 3 main regions of interest. The

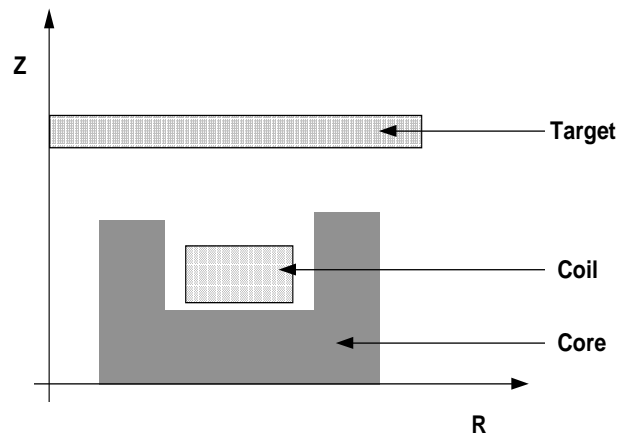


Figure 3.1 Typical model of an induction sensor

coil provides an AC driving current that generates a magnetic field. The core is generally a ferrous material to help direct the field. The target can be either a ferrous or conducting material (or both), and its proximity to the coil will affect the magnetic field. This creates a change in back EMF experienced by the coil, which can then be detected. If the magnetic properties of the target are known, the distance of the target could be estimated, or if the distance were known, the magnetic properties of the target could be calculated. This tutorial will be used to calculate the change in back EMF of the coil as the target changes position.

Getting Started

A set of models will be created by the use of the *Design Environment* to model variations of the inductive sensor. Before starting a new directory, *sensor_run1*, will be created which will hold all the files for this analysis using the *sensor* Design Environment Module. This will help to keep the files together for easier access later, should any particular file need to be used again.

UNIX Systems

The new directory can be created using

```
> mkdir sensor_run1  
> cd sensor_run1
```

Windows

The start folder can be changed using the OPERA Console Window, selecting

```
OPERA-2d ↓
    Design Environment → Change Project Folder
```

and typing the **name and path** of the new folder, *sensor_run1*, into the input box.

All Systems

Start the *Design Environment User System*.

To load the Design Environment Module for the sensor, select

```
FILE ↓
    Load DEM file
```

Change the directory within the file list box to access the *work/examples/2d* sub-directory of the installation directory, and then select the file *sensor.dem* from the list.

Running analyses

The Design Environment has the facility for generating multiple variations of a device and analysing them off-line. This is done through the generation of a *Control Set*. This control set contains a list of the variations that are to be generated, including the variable values for each variation and an associated name.

The control set name must be given first. Select

```
ANALYSIS ↓
    Set control name
```

Filename =	run1
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Two files are created based around this name.

- The control set command script: *run1.comi*

This file contains the list of commands that can be executed by the pre and post processor. These commands access other command scripts that are generated for each model variation that is created within the control set.

- The control set results file: *run1.csr*

This file is a standard file to which RESULTS are written by the post processing command scripts.

Generating variations

The menus available to configure variations of the device are found under

ANALYSIS ↓

Set **PART** data

For this analysis run we will change the geometry of the core to 2 different settings, in each case varying the size of the central hole through the core. For both of these variations we will model the target at 3 different distances from the core.

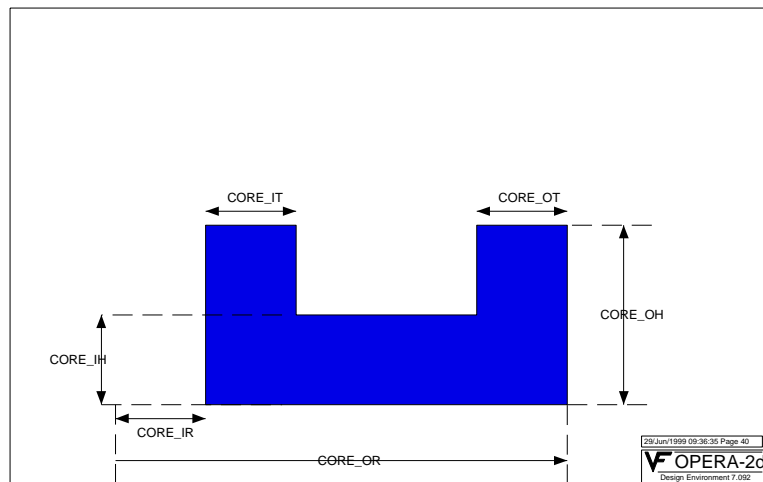


Figure 3.2 Dimensions for the core

First, change the core so that there is no hole at the centre. To do this select

ANALYSIS ↓

Set **PART** data → Core

Core Dimensions	
Inner radius	<input type="text" value="0"/>
Outer radius	<input type="text" value="10"/>
Inner flange thickness	<input type="text" value="4"/>
Outer flange thickness	<input type="text" value="2"/>
Inner height	<input type="text" value="2"/>
Outer height	<input type="text" value="4"/>

Core material properties	
IRON (Isotropic):	default.bh
Variation filename	<input type="text"/>
Start <input type="text"/>	Finish <input type="text"/> Increment <input type="text"/>
<input type="button" value="Store"/>	<input type="button" value="Update"/> <input type="button" value="Quit"/>

Fill in the dialogue box as shown and select **Update**.

Select **Quit** to leave the modifications to the core. The model should now change to reflect the changes as shown in Figure 3.3.

Now we will vary the distance of the target from the core.

Select

ANALYSIS ↓

Set **PART** data → **Target**

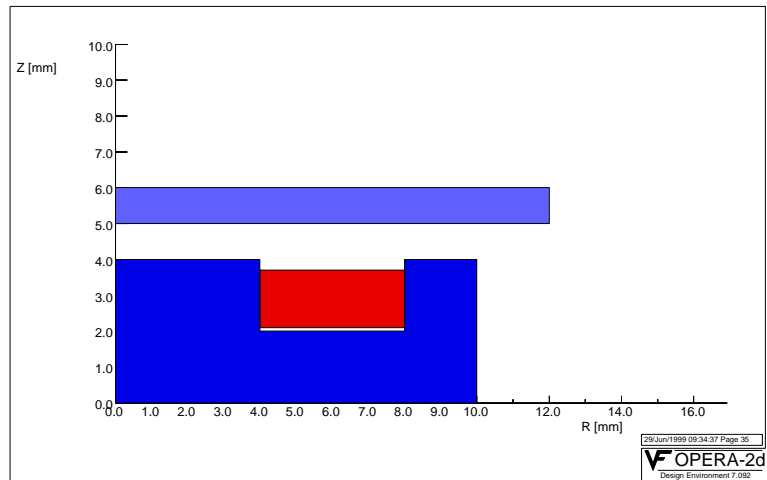


Figure 3.3 Image of the Induction sensor

Target Position		
Distance to core	<input type="text" value="*"/>	
Target Dimensions		
Thickness	<input type="text" value="1"/>	
Radius	<input type="text" value="12"/>	
Target properties		
Conductivity	<input type="text" value="5E4"/>	
FERRITE (Isotropic):	<input type="text" value="default.bh"/>	
Variation filename	<input type="text" value="radius0"/>	
Start	<input type="text" value="0.3"/>	Finish <input type="text" value="0.9"/>
		Increment <input type="text" value="0.3"/>
<input type="button" value="Store"/>		<input type="button" value="Update"/>
		<input type="button" value="Quit"/>

The use of the * for the 'Distance to Core' parameter means that this variable is to use a range of values, from **Start** to **Finish**, increasing in value by **Increment**.

Select the **Store** button followed by **Quit**. This creates files containing the model variation data. As we have a range of variable values for the distance to the target, we will create a range of model variation files. Each model variation filename is generated by appending a number to the filename stem. Therefore the three files, *radius0_1.var* to *radius0_3.var*, are created.

Now we will again modify the core so that there is a 1 mm hole at the centre. Select

ANALYSIS ↓

Set **PART** data → **Core**

Core Dimensions

Inner radius.....

1

Outer radius.....

10

Inner flange thickness

3

Outer flange thickness

2

Inner height.....

2

Outer height.....

4

Core material properties

IRON (Isotropic):

default.bh

Variation filename

Start

Finish

Increment

Store

Update

Quit

Fill in the dialogue box as shown and select **Update**.

Select **Quit** to leave the modifications to the core. Again we will vary the distance of target to the core. Select

ANALYSIS ↓

Set **PART** data → **Target**

Target Position		
Distance to core	<input type="text" value="*"/>	
Target Dimensions		
Thickness	<input type="text" value="1"/>	
Radius	<input type="text" value="12"/>	
Target properties		
Conductivity	<input type="text" value="5E4"/>	
FERRITE (Isotropic):	<input type="text" value="default.bh"/>	
Variation filename	<input type="text" value="radius1"/>	
Start	<input type="text" value="0.3"/>	Finish <input type="text" value="0.9"/>
		Increment <input type="text" value="0.3"/>
<input type="button" value="Store"/>	<input type="button" value="Update"/>	<input type="button" value="Quit"/>

and complete the dialogue box as shown below and select **Store**. This creates another five model variations, *radius1_1.var* to *radius1_3.var*.

Preparing and Analysing Models

We have created six model variations. All of these can be converted into OPERA-2d data files, analysed and post processed by selecting

ANALYSIS ↓

Prepare and run analysis

This menu option:

- Loads each model variation, and prepares it for analysis and post processing. In doing this an OPERA-2d data file (extension *.op2*) is created, together with a post processing command script file (extension *.script*) containing the commands necessary for solving and post processing this particular file.
- Prepares the Control set command script file to access each of the model variations that is included

- Starts the OPERA-2d pre and post processor¹, and begins the analysis and post processing of all the models.

This command may take some time to execute, and generally the user would be advised to run this command at a time when the analysis of the models will not adversely affect users of the computer system, for example it could be started to run overnight

An alternative to this option is to select **Prepare control set** from the menu. This will create all necessary files to run the analysis. The analysis can then be started by selecting the **Run analysis**, or the pre and post processor can be started and the control set command script run by selecting the file *run1.comi* through

```
FILE↓
    Load a command file
```

Studying results

After analysis, any values stored using the **RESULTS** post processing command, in this case back EMF, can be viewed. This is done using:

```
ANALYSIS ↓
    View analysis results
```

and the file *run1.csr* selected from the file list box. The results from all analyses are listed in a message box.

Individual results are also stored with each model variation. Any of the model variations can be loaded, setting the values of variables to those used for this variation. This is done by selecting:

```
ANALYSIS ↓
    Load model variation
```

and the name of the file selected. For example selecting *radius0_2.var* will load the model variation with no hole at the centre of the core, and the target at 0.6 mm from the core. The results for this particular analysis are displayed when loading the model.

1. Note for UNIX systems the launching of the pre and post processor requires that the alias OPERAPP be defined within the UNIX shell.

Chapter 4

Induction Sensor Configuration

Induction Sensor

Figure 4.1 shows a typical design for an induction sensor.

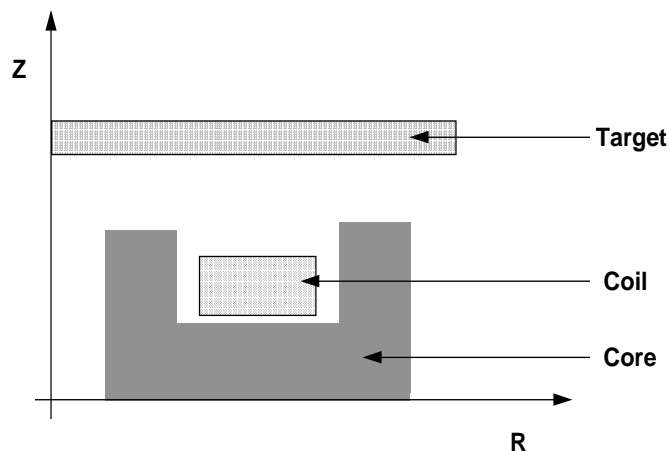


Figure 4.1 A Typical Induction Sensor

The device is axisymmetric in nature with 3 main regions of interest. The coil provides an AC driving current that generates a magnetic field. The core is a ferrous material to help direct the field. The target can be either a ferrous or conducting material, and its proximity to the coil will affect the magnetic field. This creates a change in impedance in the coil, which can be measured. If the magnetic properties of the target are known, the distance of the target can be estimated, or if the

distance is known, the magnetic properties of the target can be calculated from this measurement.

This tutorial will demonstrate how a Design Environment Module (DEM) of this device can be generated. The chapter “[Induction Sensor User System](#)” on page 3-1 shows how this file is subsequently used by a user.

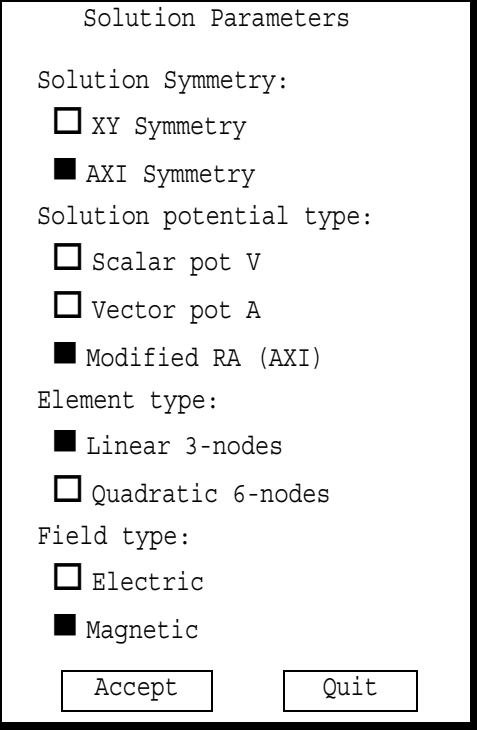
Launch the *Configuration System*.

First the solution parameters and units must be set to the correct values for modelling of a magnetic axisymmetric device. This is done by:

MODEL ↓

Solution settings

Complete the dialogue box as follows and then **Accept**

A dialog box titled "Solution Parameters" with a black border. It contains several sections with labels and checkboxes. The "Solution Symmetry" section has "XY Symmetry" (unchecked) and "AXI Symmetry" (checked). The "Solution potential type" section has "Scalar pot V" (unchecked), "Vector pot A" (unchecked), and "Modified RA (AXI)" (checked). The "Element type" section has "Linear 3-nodes" (checked) and "Quadratic 6-nodes" (unchecked). The "Field type" section has "Electric" (unchecked) and "Magnetic" (checked). At the bottom are two buttons: "Accept" and "Quit".

Solution Parameters

Solution Symmetry:

☐ XY Symmetry

☒ AXI Symmetry

Solution potential type:

☐ Scalar pot V

☐ Vector pot A

☒ Modified RA (AXI)

Element type:

☒ Linear 3-nodes

☐ Quadratic 6-nodes

Field type:

☐ Electric

☒ Magnetic

Accept Quit

Select **Return** to go to the top menu bar.

The model units must be set using the following sequence. Firstly the length unit is set to millimetres:

```
UNITS ↓  
    Length unit → Millimetre
```

and **Return**.

Secondly, the conductivity unit is set:

```
UNITS ↓  
    Conductivity unit → Siemen/mm
```

and **Return**.

Finally, the current density unit is set:

```
UNITS ↓  
    Density unit → Amps/mm**2
```

and **Return**.

Select **Return** again to return to the top level menu.

The Core

Dimension Scheme

Firstly the design for the whole model should be considered and planned with respect to the requirements of the model and the users. The three main regions will each be entered separately.

Start with the modelling and dimensioning of the core. Figure 4.2 shows the dimension scheme that will be applied to the core.

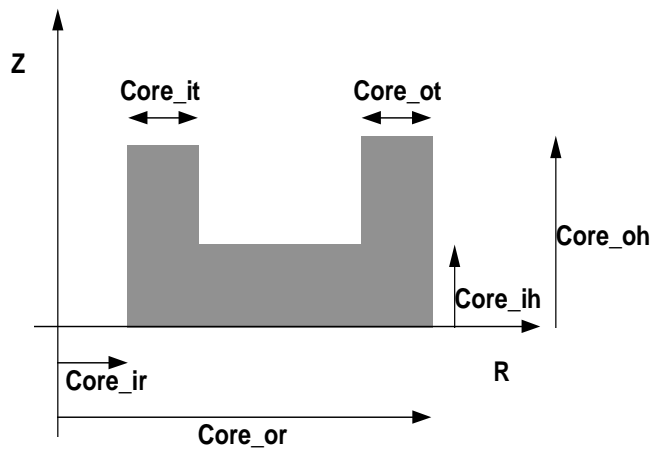


Figure 4.2 Dimension scheme for the core

In this case 6 variables will be used to define the different dimensions that are going to set the size and shape of the core. The model being considered is quite simple, with certain assumptions being made so that only these few variables need be defined.

- The inner and outer flange will both be assumed to be the same height (**core_oh**)
- All faces of the core will be either vertical or horizontal - there will be no angled surfaces and no curved surfaces.
- There will be no chamfers to smooth the corners

Design Variables

The variables that will be used in constraining the core's geometry must now be defined. This is done using the **VARIABLE** command. In each case the variable name will be given, followed by a value to which the variable is to be set and a description of the variable for later use in the *User System*.

The first variable to be set is **Core_ir**.

MODEL ↓

Variables → Define new variable

Variable name =	Core_ir
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Complete and **Accept** the following dialogue box:

Expression =	2
Descriptor =	Inner radius
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The next variable to be set is **Core_or**.

MODEL ↓

Variables → Define new variable

Variable name =	Core_or
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Expression =	10
Descriptor =	Outer radius
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The remaining four variables, their values and descriptions, are listed below. Repeat the menuing used for the previous two variables to define these variables as well:

Variable name	Expression	Descriptor
Core_it	2	Inner flange thickness
Core_ot	2	Outer flange thickness
Core_ih	2	Inner height
Core_oh	5	Outer height

To show a list of all the variables that have been defined so far:

MODEL ↓
Variables → List variables

A list of all the variables defined with their expressions is displayed. The value given to any variable can be easily changed. In this case **Core_oh** will be modified to the new value of 4. Complete the following menuing:

MODEL ↓
Variables → Modify variable

From the list of 6 variables available, select **CORE_OH** and **Accept** and then enter the new value for this variable and **Accept**.

Expression =	4
Descriptor =	Outer height
<div>Accept</div> <div>Dismiss</div>	

All the variables used to define the core have now been defined.

Geometry and constraints

A polygon must now be defined to model the core. This is done using the **POLYGON** command which only has a single parameter - the number of sides in the polygon, in this case 8.

MODEL ↓
Geometry → New polygon

Number of polygon sides =	8
<div>Accept</div> <div>Dismiss</div>	

An octagon appears centred in the screen.

It is now necessary to constrain the geometry to fit the dimension scheme shown earlier, and so choose **Return** to return to the main modelling menu options.

The first point in the polygon, labelled 1.01, will be constrained to lie on the R axis, at the radius of **Core_ir**. This is done using the **POINT** constraint by entering

MODEL ↓
Constraints → By keyboard → Fixed point

Complete the dialogue box as shown and then **Accept**

Point name =	1.01
X position =	Core_ir
Y position =	0
<div>Accept</div> <div>Dismiss</div>	

and the geometry will change to position point **1.01** at (2,0).

The second point of polygon one, point **1.02**, will be constrained by

MODEL ↓

Constraints → **By keyboard** → **Fixed point**

Complete the following dialogue box as shown and then **Accept**

Point name =	1.02
X position =	Core_or
Y position =	0
<div>Accept</div> <div>Dismiss</div>	

This will position the second point on the R axis at (Core_or,0). The other points are all going to be positioned relative to these 2 points, so **VECTOR** constraints will be used.

Point **1.03** is directly above **1.02** at a height of **Core_oh**, and point **1.04** lies directly inside point **1.03**.

MODEL ↓

Constraints → **By keyboard** → **Vector**

Complete the dialogue box as shown and then **Accept**

First constrained point =	1.02
Second constrained point =	1.03
X vector =	0
Y vector =	Core_ih
<div>Accept</div> <div>Dismiss</div>	

The incorrect value for the y-component of this constraint has been set and so this constraint needs modifying. Firstly, the constraint number needs to be identified. Complete the following menuing to list the constraints used by polygon 1:

MODEL ↓

Constraints → Polygon points

Complete the dialogue box as shown and then **Accept**:

Start from polygon =	1
Finish at polygon =	1
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

This shows the constraint numbers, constraint types¹, point names and expressions used by polygon 1. The incorrect constraint expression was applied between points **1.02** and **1.03**. From the data shown it can be seen that constraint number 6 was used to define the y-component of the vector between points **1.02** and **1.03**. The value **Core_ih** must be changed to **Core_oh**. The constraint is modified as follows:

MODEL ↓

Constraints → Modify constraint

Complete the following dialogue boxes and **Accept**

Constraint number =	6
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

A message box will appear identifying constraint 6 with its old expression. A dialogue box will appear which needs to be completed as below:

New expression =	Core_oh
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The core geometry should now change and reflect the constraint with its new value. If the wrong point name had been used instead of a wrong expression, the constraint would need to be deleted and re-defined with the correct point names. There is no means of modifying or editing constraint point names.

Further constraints that need to be applied to the core can now be defined.

MODEL ↓

Constraints → By keyboard → Vector

-
1. The vector and point constraints have been separated into 2 distinct constraints - one for the X component and one for the Y component.

Complete the dialogue box as shown and then **Accept**

First constrained point =	1.04
Second constrained point =	1.03
X vector =	Core_ot
Y vector =	0
<div>Accept</div> <div>Dismiss</div>	

(NB: Note the vector going *from 1.04 to 1.03* is positive)

Note: It may be the case that a menu obscures some of the required model. The menus can be temporarily hidden by selecting F1. Pressing F1 a second time makes the menus visible again.

Similar constraints can be applied to fix **1.08** and **1.07** relative to **1.01**. The points and constraints listed below need to be set in the model using the same menuing as described above.

Constraint	First point	Second point	X component	Y Component
VECTOR	1.01	1.08	0	Core_oh
VECTOR	1.08	1.07	Core_it	0

Choose **Return** twice to return to the top level menu. To see the status of the constraints so far we can use the **LIST** command.

MODEL ↓

Geometry → List polygon points

Complete the dialogue box as shown and then **Accept**

Start from polygon =	1
Finish at polygon =	1
<div>Accept</div> <div>Dismiss</div>	

and a list of points for polygon 1 are shown. From this it can be seen that only points **1.05** and **1.06** are still to be constrained. By adding the following constraints with the menuing system, the polygon becomes fully constrained.

MODEL ↓

Constraints → By keyboard → Vector

Constraint	First point	Second point	X component	Y Component
VECTOR	1.04	1.05	0	Core_ih-Core_oh
VECTOR	1.07	1.06	0	Core_ih-Core_oh

The Core is now fully constrained. The full list of constraints that can be applied can be seen by

MODEL ↓

Constraints → Polygon points

Complete the dialogue box as shown and then **Accept**

Start from =	1
Finish at =	1
<div>Accept</div> <div>Dismiss</div>	

To assist the DEM user, we will set limits to prevent values being entered that would create a geometric error. Limits to be set are

$$\begin{aligned}
 &Core_ih > 0 \\
 &Core_ih \leq Core_oh \\
 &Core_ir \geq 0 \\
 &Core_it \geq 0 \\
 &Core_ot \geq 0
 \end{aligned}
 \tag{4.1}$$

This is done using the limits command A strictly positive condition is applied with:

MODEL ↓

Variables → Limit variable values → Apply positive limit

Complete the dialogue box as shown and then **Accept**

Limit Variable Expression	
Expression =	Core_ih
Accept	Quit

:Other conditions can be more generally applied using:

MODEL ↓

Variables → Limit
variable values → Limit variable expression

Complete the dialogue box as shown and then **Accept**:

Limit Variable Expression	
Expression 1	<input type="text" value="Core_ih"/>
<input type="checkbox"/> LT <input checked="" type="checkbox"/> LE <input type="checkbox"/> GT <input type="checkbox"/> GE <input type="checkbox"/> EQ <input type="checkbox"/> NE	
Expression 2	<input type="text" value="Core_oh"/>
Associated message	<input type="text" value="Heights are incorrect"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

The other three limits listed below can be set using the same menuing:

First exp.	Operator	Second exp.	Associated message
Core_ir	ge	0	Inner radius is too small
Core_it	ge	0	Inner flange is too thin
Core_ot	ge	0	Outer flange is too thin

A list of all the limits that have been specified can be seen by:

MODEL ↓

Variables → Limit variable values → List limits

Choose **Return** three times to return to the top level menu options.

Material Properties

The material properties of the region should now be set. The core will be non-conducting and will normally be run as a non-linear problem, so permeability need not be parameterized. We therefore set the material properties as

- A non-linear material, using the default material name **IRON**
- Zero Conductivity
- A permeability of 1000 (for linear testing purposes)
- A subdivision size based on the outer radius and height

This is done by

MODEL ↓

Material data → By keyboard

Polygon number =	1
Accept	Dismiss

Complete the following dialogue box and **Accept**

Set the material properties	
<input checked="" type="checkbox"/> Polygon	<input type="checkbox"/> Background
AIR, CONDUCTOR, IRON	IRON
Maximum subdivision	(Core_oh+Core_or)/30
Permeability	1000
Conductivity	0
Current density	0
Phase of the region	0
Velocity of region	0
Conductor number	0
<input checked="" type="checkbox"/> Connected	<input type="checkbox"/> Disconnected
Include region in OP2 file	1
Accept	Quit

Choose **Return** twice to return to the top level menu options. The Core has now been defined.

The data created can be stored to a file at any time by:

FILE ↓

Save as new DEM file

Filename =	core.dem
Accept	Dismiss

to create a new DEM file.

The Coil

Dimension Scheme

The coil is a simple rectangular shape. It is to be positioned in the inner section of the core, and will be fully defined by 4 parameters, shown in Figure 4.3.

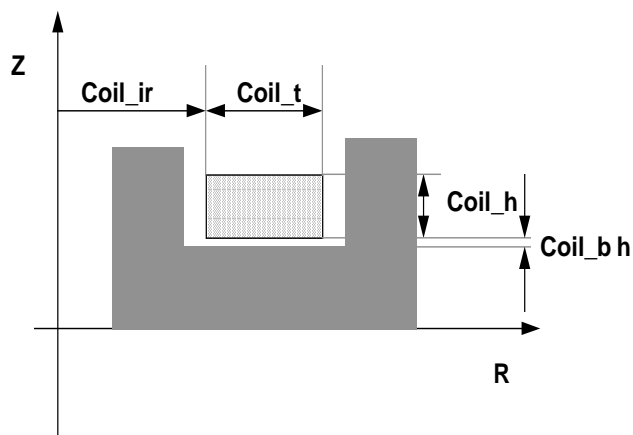


Figure 4.3 Dimension scheme for the coil

Design Variables

The 4 variables that are to be used are defined by

MODEL ↓

Variables → Define new variable

Variable name =	Coil_ir
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Complete the following dialogue box and **Accept**

Expression =	4
Descriptor =	Inner radius
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

The other three variables that need to be defined using the identical menuing are:

Variable name	Expression	Descriptor
Coil_bh	0.1	Base height
Coil_t	4	Radial thickness
Coil_h	1.6	Height

Geometry and Constraints

As before, the coil is created by

MODEL ↓

Geometry → New polygon

Number of polygon sides =	4
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

to create a quadrilateral. Choose **Return** to return to the main modelling options. The new polygon must now be constrained. The first point of polygon 2, labelled 2.01, is constrained to lie at fixed coordinates by

MODEL ↓

Constraints → By keyboard → Fixed point

Point name =	2.01
X position =	Coil_ir
Y position =	Core_ih+coil_bh
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

This fixes the position of the bottom inside corner of the coil. Other points will be positioned relative to this point using vector constraints.

MODEL ↓

Constraints → By keyboard → Vector

First constrained point =	2.01
Second constrained point =	2.02
X vector =	Coil_t
Y vector =	0
<div>AcceptDismiss</div>	

The other points listed below can be similarly constrained:

Constraint	First point	Second point	X component	Y component
VECTOR	2.01	2.04	0	Coil_h
VECTOR	2.02	2.03	0	Coil_h

The coil is now fully constrained. Choose **Return** twice to return to the main modelling options.

To prevent the DEM user entering bad dimensions, for example by setting variable values such that the coil overlaps the core, more limits will be entered.

MODEL ↓

Variables → Limit variable values → Limit variable expression

Complete the following dialogue box and **Accept**

Limit Variable Expression	
Expression 1	Coil_h
<input type="checkbox"/> LT <input type="checkbox"/> LE <input checked="" type="checkbox"/> GT <input type="checkbox"/> GE <input type="checkbox"/> EQ <input type="checkbox"/> NE	
Expression 2	0
Associated message	Height must be positive
Accept	Quit

The following limits can be entered using the same menuing:

Expression	Operator	Expression	Description
Coil_t	gt	0	Thickness must be positive
Coil_bh	ge	0	Base height is incorrect
Coil_ir	ge	Core_ir+Core_it	Inside of the coil overlaps the core
Coil_ir+Coil_t	le	Core_or-Core_ot	Outside of the coil overlaps the core
Core_or-(Core_ir+Core_it+Core_ot)	ge	Coil_t	The coil is wider than the core slot

Material Properties

Although modelled as a single region, the coil is made from a group of windings. The design should include such data within it. The data needed is

- Number of turns
- Current per turn
- Wire radius
- Resistivity of the wire

We can therefore create variables to represent these values, with default values. Their value will be set in the *User system*, so they are given descriptions that will be used as part of the menuing structure to give the user a clear meaning for the variable values.

MODEL ↓

Variables → Define new variable

Variable name = **Wire_d**

Expression = **0.4**
 Descriptor = **Wire diameter**

The rest of the coil winding variables are:

Variable name	Expression	Descriptor
Cur_tur	0.1	Current / turn
Num_tur	30	Number of turns
Wir_rho	16e-6	Wire resistivity (S/mm)

The material properties will use the number of turns and current / turn to calculate the current density, and the material will be set to a conductor. The subdivision size will be based on the coil size.

MODEL ↓

Material data → By cursor → Pick region

Click the mouse cursor in the region area (6,3).

The variable **AREA** can be used in material property variables and is calculated when needed, with the correct value of the region's area. NB, this variable may not be used as part of a constraint expression. It is restricted to expressions involving material characteristics.

Set the material properties

☒ Polygon
 ☐ Background

AIR, CONDUCTOR, IRON CONDUCTOR

Maximum subdivision (Coil_h+Coil_t)/15

Permeability 1

Conductivity 0

Current density (Num_tur*Cur_tur)/AREA

Phase of the region 0

Velocity of region 0

Conductor number 0

☒ Connected
 ☐ Disconnected

Include region in OP2 file 1

Accept
Quit

A physical limit will be added to ensure that the area taken by the windings is not greater than the area of the region representing the coil. Choose **Return** to get back to the main modelling options and then:

MODEL ↓

Variables → Limit

variable values → Limit variable expression

Complete the following dialogue box and **Accept**

Limit Variable Expression	
Expression 1	<input type="text" value="Num_tur*(Wire_d**2)"/>
<input type="checkbox"/> LT <input checked="" type="checkbox"/> LE <input type="checkbox"/> GT <input type="checkbox"/> GE <input type="checkbox"/> EQ <input type="checkbox"/> NE	
Expression 2	<input type="text" value="Coil_h*Coil_t"/>
Associated message	<input type="text" value="Coil cross-section too small"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

Some of these variables will also be used later when preparing the post processing command script.

The Target

Dimension Scheme

The target is also a simple rectangular shape. It is to be positioned above the core as shown in Figure 4.4

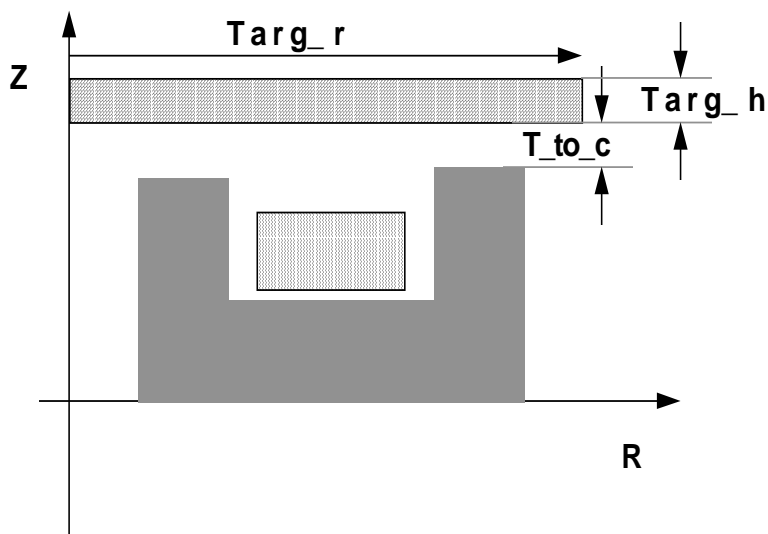


Figure 4.4 Dimension scheme for the target

Design Variables

The 3 variables that are to be used are defined by:

MODEL ↓

Variables → Define new variable

Variable name =	Targ_r
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Expression =	12
Descriptor =	Radius
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The other variables associated with the target are listed below and can be defined using the menus described above.

Variable name	Expression	Descriptor
Targ_h	1	Thickness
T_to_c	1	Distance to the core
Targ_c	5e4	Conductivity

The final variable defined, *Targ_c*, is to be used in defining the material properties, so that the conductivity of the region can be easily modified by the user. Select **Return** to return to the main modelling menu once the variables have been defined.

Geometry and Constraints

Polygon 3, used to represent the target is created and constrained by:

MODEL ↓

Geometry → New polygon

Number of polygon sides =	4
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The first point is constrained as a point position by:

MODEL ↓

Constraints → By keyboard → Fixed point

Complete the following dialogue box and **Accept**

Point name =	3.01
X position =	0
Y position =	Core_oh+T_to_c
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The other points can be constrained as vectors

MODEL ↓

Constraints → By keyboard → Vector

First constrained point =	3.01
Second constrained point =	3.02
X vector =	Targ_r
Y vector =	0

Accept

Dismiss

The other points are constrained as follows:

Constraint type	First point	Second point	X component	Y component
VECTOR	3.01	3.04	0	Targ_h
VECTOR	3.02	3.03	0	Targ_h

Limits are added to prevent the target colliding with core or coil

MODEL ↓

Variables → Limit

variable values → Limit variable expression

Limit Variable Expression	
Expression 1	Targ_h
<input type="checkbox"/> LT <input type="checkbox"/> LE <input checked="" type="checkbox"/> GT <input type="checkbox"/> GE <input type="checkbox"/> EQ <input type="checkbox"/> NE	
Expression 2	0
Associated message	Target requires a thickness
Accept	Quit

The following limits can also be entered using the same menuing:

First expression	Operator	Second expression	Message
Targ_r	gt	0	The target requires a radius
T_to_c	ge	0	The target overlaps the core
Core_ih+Coil_bh+Coil_h	le	Core_oh+T_to_c	The target overlaps the coil

Material Properties

The target is a conducting, ferrous type material. It will be given a value for conductivity, a default value for permeability, and be assigned a non-linear material type called **FERRITE**. A BH data label called **FERRITE** is then generated so that a BH curve can be assigned to this material type.

MODEL ↓

Material data → By keyboard

Polygon number = **3**

Accept
Dismiss

Set the material properties

☒ Polygon
 ☐ Background

AIR, CONDUCTOR, IRON	FERRITE
Maximum subdivision	Targ_r/20
Permeability	1000
Conductivity	Targ_c
Current density	0
Phase of the region	0
Velocity of region	0
Conductor number	0
<input checked="" type="checkbox"/> Connected <input type="checkbox"/> Disconnected	
Include region in OP2 file	1

Accept
Quit

As the target is a ferrous type material, it requires a suitable BH curve to describe its magnetic characteristic. In this example, we shall use the default curve that is attributed to material **IRON** as the basis of the **FERRITE** BH curve. Hence select:

MODEL ↓

BH or DE data → IRON → Store in file

and fill in the dialogue box as follows and **Accept**

Data filename =	Default
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

select **Return** and then select **FERRITE** followed by **Load from file**

Choose the BH data file *default.bh* from the list and **Accept**

The BH data has now been assigned to material FERRITE.

Select **Return** twice to close the BH data menuing. The target has now been defined and so select **Return** to return to the main modelling menu options To redraw the model, select

DISPLAY ↓
Refresh

The Background

Dimension scheme

The final region that must be added is the background air. This will stretch out well beyond the rest of the model. Firstly we will define variables to represent the maximum size of the model, and use these to position points within the model.

MODEL ↓

Variables → Define new variable

Variable name =	Max_h
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Expression =	Core_oh+T_to_c+Targ_h
Descriptor =	
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

MODEL ↓

Variables → Define new variable

Variable name =	Max_r
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Expression =	(targ_r+Core_or)/2
Descriptor =	
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

No description was included with these variables as they will never be set directly in the *User System*. These values are always calculated from the expressions given above, using the other variables that are set in the *User System*.

Geometry and constraints

Polygon 4 is created to represent the air region around the model, and its points are constrained

MODEL ↓

Geometry → New polygon

Number of polygon sides =	4
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Select **Return** to return to the main modelling menu options.

MODEL ↓

Constraints → By keyboard → Fixed point

Point name =	4.01
X position =	0
Y position =	-4*Max_h
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The other points are also constrained as follows:

Constraint	First point	Second point	X component	Y component
VECTOR	4.01	4.02	$6*Max_r$	0
FIXED POINT	4.04	none	0	$5*Max_h$
VECTOR	4.04	4.03	$6*Max_r$	0

Material properties

The shape type needs to be changed so that the system recognises that it is a *Back-ground* region. The default properties of regions are those of air so the subdivision size is the only other parameter that must be set. This is done by

MODEL ↓

Material data → By cursor → Pick region

Click the mouse cursor in the region area (10,10)

Set the material properties

☐ Polygon ☒ Background

AIR, CONDUCTOR, IRON

Maximum subdivision

Permeability

Conductivity

Current density

Phase of the region

Velocity of region

Conductor number

☒ Connected ☐ Disconnected

Include region in OP2 file

This completes the initial model definition for the inductive sensor. The model is saved using

FILE ↓

Save as new DEM file

Filename = **sensor.dem**

to create a new DEM file.

Boundary Conditions

For this model, the only boundary condition that need be applied is to set the vector potential along the Z axis to be zero. By setting the background region face only, any face that intersects it, such as the face at the centre of the target, will automatically have the condition transferred onto it, when creating the data files for analysis.

This boundary condition is set by

MODEL ↓

Boundary conditions

Complete the following dialogue box and **Accept**

Set boundary condition

Type of condition:

☒ V
 ☐ DV
 ☐ CLEAR
 ☐ SYMM

Expression for potential value:

Value

Continue

A message box appears stating:

Adding the boundary condition: F=V, V=0

Click on **Continue** or press any key on the keyboard to clear it.

From the menu that appears, choose **By cursor**

The system is now waiting for sides to be selected with the mouse. Click the left mouse button at a point near the left hand side of the background air region, e.g.

<mouse> (0.0,0.0)

Select **Return** to accept the selected side and to set the boundary condition on it.

Checking the model

The model data set up in the Configuration System is now complete and able to generate pre processing files. Within the configuration System we are able to check the validity of the OPERA-2d data that will be created and analysed.

From the menus select

```
MODEL ↓
    Check OPERA-2d data
```

For larger models this may take some time as the model data is converted into the OPERA-2d data format. A message box appears listing some checks that have been performed. There should be no errors and no warnings reported in this message box. Select **Continue**.

We can generate the mesh that will be created by the model by selecting

```
MODEL ↓
    Check OPERA-2d data → Generate mesh
```

The finite element mesh is generated, and a message box detailing some information about the mesh and model is shown. Select **Continue** to clear this message box.

The mesh can be viewed by selecting

```
Check OPERA-2d data → View Pre-
                      Processor model → +Mesh(-Mesh toggle)
Check OPERA-2d data → View Pre-
                      Processor model → Refresh
```

Similarly, the boundary conditions applied can be checked by selecting

```
Check OPERA-2d data → View Pre-
                      Processor model → Nodes → Boundary
                                                nodes
Check OPERA-2d data → View Pre-
                      Processor model → Refresh
```

The region data can also be checked using the options under

```
Check OPERA-2d data → Print Pre-Processor data
```

Setting analysis options

The analysis module and options to be used by the analysis must be set. In this example we are using the AC analysis module with the default frequency of 50Hz, with non-linear material properties and mesh refinement.

Select

MODEL ↓
Analysis data → Steady state
harmonic (AC) → Non-linear analysis

Leave the default non-linear options of **21** iterations and a convergence tolerance of **0.001**, and select **Accept**. The mesh generated by the region subdivision settings are generally good, but we will try to allow the solver to refine the mesh to obtain a better mesh if necessary. To do this select

MODEL ↓
Analysis data → Steady state
harmonic (AC) → Mesh refinement options

Maximum number of iterations =	=	3
Maximum number of elements =	=	*
Final convergence accuracy(%)	=	1
<div>Accept</div>		<div>Dismiss</div>

to allow a maximum of 3 mesh refinements, and **Accept**.

These settings can be changed later in the *User System* if necessary, for example, if modelling different drive frequencies.

Setting the Post Processing

An important feature of the *Design Environment* is the ability to set up automatic post processing that will run through a series of pre-determined commands and process a particular result. The post processing for the induction sensor will involve the calculation of the back EMF, using the calculation of power loss in the target. The result will be written to a file, so that the post processing can be run in batch mode and the user will be able to retrieve results after all the analyses have been executed.

The first command to be added is an area integral over the target region. This is achieved by:

DEM ↓

Post processing → Add command

and from the list of post processing commands available select **INTA**.

Complete the following dialogue box and **Accept**

INTA command			
Component	POT		
Region or material	3	Time around AC Cycle	TAVE
Accept		Dismiss	

The post processing command **INTA** calculates several useful quantities, including the average power loss of the target. This result is placed in the system variable **POWER**. This power, along with the RMS current in the coil, is used to calculate the back EMF experienced by the coil:

Select **\$ CONSTANT** from the list of commands and fill in the following dialogue box:

\$ CONS command	
Variable name	#EMF
Set value	POWER/(Cur_tur/SQRT(2))
Accept	Dismiss

The user variable **#EMF** holds the RMS back EMF experienced by the coil. This value can then be written automatically to a text file by selecting the **RESULTS**

command. This command writes data in a standard format into a file for access later by the user. Complete the dialogue box as follows and **Accept**

RESULTS command	
Description	Back EMF value
Value	#EMF
<div>Accept Dismiss</div>	

Choose **QUIT** to leave the *add commands* menu, and select **Show command file**, which will list out the command lines that will be used to run the post processor. This set of command lines will be generated independently for every OPERA-2d data file that is created. The commands are placed in a file with the same name as the OPERA-2d data file, but with a *.script* extension rather than *.op2*. Select **Return** twice to shut the post processing menus down.

The updated model data should now be saved into the *.dem* file. This is done by the following menu option:

FILE ↓
 Save DEM file

This stores the model in the last filename used, *sensor.dem*.

Preparing PARTS for the User System

The model is now complete and various parameters can be changed, and the model will vary accordingly. However, the system is not easy to use at this time as the names of all variables must be known before they can be changed.

To assist another user of the software the *User System* is available. This has only restricted options and specially prepared menus and views, unique to the model that has been built. These views and menus must be set up in the *Configuration System*. This is done by creating PARTS. Three parts will be created to make the setting of parameters easier. The parts for this model will be called **CORE**, **COIL** and **TARGET** to represent the three physical sections of the model.

The Core

Firstly, the display should be set to the view that the *User System* will display when variables for the Core are being set. The size of the screen will be stored as expressions so that changes in dimension can be reflected in the part view. New variables will be defined for expressions that will be repeated in the following section.

MODEL ↓

Variables → Define new variable

Complete and **Accept** the following dialogue boxes.

Variable name = **core_s**

Accept
Dismiss

Expression = **Core_or*1.5**
 Descriptor =

Accept
Dismiss

The following variable can also be set using the same menuing as described above:

Variable name	Expression	Descriptor
Core_g	Core_s/12	

Once these variables have been set, select Return twice to return to the top menu bar. The view that the part shows can now be set. Select:

DISPLAY ↓

Hide labels (*Label polygons toggle*)

DISPLAY ↓

Options

and then complete the following dialogue box and **Accept**

Refresh the viewing screen

Display border

X minimum

X maximum

Y minimum

Y maximum

Show nodes:

☒ NONE ☐ ALL ☐ REFE ☐ POLY

Show variable names:

☒ NONE ☐ ALL ☐ SAME

+ or -

Show polygon:

☒ NONE ☐ ALL ☐ SAME

+ or -

DISPLAY ↓

Refresh

Choose **Return** from the menu. The view now shows only the core. This view is stored with the part that is created by:

DEM ↓

Create or modify PARTS → Create new part

Complete and **Accept** the menu box

Part name = **Core**

The system is now waiting for different items that should be associated with this part. The options available are:

- **List Part commands** - This allows you to list the commands that have been associated with the part.
- **Add part command** - This allows you to add a command to the list of commands associated with the part.
- **Delete part command** - This allows you to delete a command from the list of commands.
- **Insert part command** - This allows you to insert a command in at a point within the command list.
- **Quit** - This is allows you to quit the part command and save the commands associated with the part.

The first thing to do is to set a text string that describes the commands associated with the part:

Add part command → Set text

Complete the dialogue box with the text string and **Accept**

Text message =	Core dimensions
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Having set some text, it is now time to state the variables whose values can be set in this part.

Select **Ask for variable**, and from the list of variables select **Core_ir** followed by **Accept**.

Repeat this for the following 5 variables:

Ask for variable → Core_or
Ask for variable → Core_it
Ask for variable → Core_ot
Ask for variable → Core_ih
Ask for variable → Core_oh

The following text is now added to the list of commands associated with the part:

Select **Set text**

Complete the dialogue box with the text string and **Accept**

Text message =	Core material properties
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The user is able to view the BH data assigned to a non-linear material.

Select **Show material BH data**

Complete the dialogue box and **Accept**

Material name	IRON
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

To list the part commands that have been defined the list option must be chosen:

Select **List part commands**

The commands associated with the part have now been defined and so the part menuing can be closed. Select **Return, Quit** and then **Return** to return to the top menu level.

The menu this creates for the part can be seen using

DEM ↓
Demonstrate a part → Core

It may also be necessary to create dimension arrows that represent the variables being set, so that the user can see the effect each variable has on the model. As the model can change size the dimension arrows must be free to move with the model. This is done by positioning the ends of the dimension arrows at *reference points*. these points can then be constrained in exactly the same way as polygon nodes were constrained earlier.

Reference points are labelled **0.001** through to **0.999**, allowing up to 999 reference points to be used in a model. Reference points are created when the name of the point is used in either a constraint command or in the variable command.

Firstly the display options need to be set up.

DISPLAY ↓
Options

Complete the dialogue box as follows and then **Accept**

Refresh the viewing screen	
Display border	
X minimum	<input type="text" value="0"/>
X maximum	<input type="text" value="10"/>
Y minimum	<input type="text" value="0"/>
Y maximum	<input type="text" value="10"/>
Show nodes:	
<input type="checkbox"/> NONE	<input checked="" type="checkbox"/> ALL
<input type="checkbox"/> REFE	<input type="checkbox"/> POLY
Show variable names:	
<input type="checkbox"/> NONE	<input checked="" type="checkbox"/> ALL
<input type="checkbox"/> SAME	
+ or -	<input type="text"/>
Show polygon:	
<input type="checkbox"/> NONE	<input checked="" type="checkbox"/> ALL
<input type="checkbox"/> SAME	
+ or -	<input type="text"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

DISPLAY ↓
Refresh

Return updates the screen and takes you to the top menu level. Now the dimension lines have to be set up so as to clearly show the effect variable changes have on the model geometry:

The reference point **0.001** will be created and constrained at a point as follows:

DEM ↓
Dimension lines → By keyboard → Fixed point

Complete the following dialogue box and **Accept**

Point name =	0.001
X position =	0
Y position =	-core_g
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The reference point 0.002 will be created by adding a vector constraint between points 0.001 and 0.002.

DEM ↓

Dimension lines → By keyboard → Vector

First constrained point =	0.001
Second constrained point =	0.002
X vector =	core_or
Y vector =	0
<div>Accept</div> <div>Dismiss</div>	

DEM ↓

Dimension lines → Define/edit dimension

The variable to be dimensioned needs to be selected from the menu list. Hence select **Core_or** from the list of variables.

Having selected the variable the various options to define the dimension need to be set. Select: **Set dimension options**

Complete the dialogue box as follows and **Accept**

Create or modify variable	
Variable Descriptor	Outer radius
Dimension label type:	
<input type="checkbox"/> NONE	<input checked="" type="checkbox"/> LINE
<input type="checkbox"/> ARC	<input type="checkbox"/> CEN
<input type="checkbox"/> CENA	<input type="checkbox"/> CENC
Dimension start point	0.001
Dimension end point	0.002
Dimension centre point	
Displacement of line	0
Expression for curvature	0
X-shift for the label	0
Y-shift for the label	0
<div>Accept</div> <div>Quit</div>	

These options define the location and type of dimensioning arrow. There are default parameters that set the arrow-head size and line thickness. The arrow-heads for this dimension are changed by choosing: **Set line style**

Complete the following dialogue box and **Accept**

Set the line parameters	
Dimension line type	<input type="text" value="0"/>
Line type of end bar	<input type="text" value="1"/>
Arrow direction:	
<input type="checkbox"/> NONE	<input type="checkbox"/> BOTH <input checked="" type="checkbox"/> FORW <input type="checkbox"/> BACK
Arrow length expression	<input type="text" value="core_g/4"/>
End bar length expression	<input type="text" value="core_g*1.5"/>
End bar shift expression	<input type="text" value="-core_g*0.25"/>
<input type="button" value="Update"/>	<input type="button" value="Accept"/> <input type="button" value="Quit"/>

The dimension arrow is now positioned correctly on the model. Any changes to the size of the model will be reflected by the dimension arrow.

Further dimension arrows can be set for the remaining variables in the part. This can be done using the same idea of constraining reference points and using these as the end points for dimension lines. This will be left as an exercise for the user.

The variables associated with the Core have now been fully dimensioned and the effect of a variable change on the core geometry is clearer to see. Having set up the Core part, it is possible to run the Core menuing options and set the variables using this. Select:

DEM ↓
Demonstrate a part → Core

The Coil

As with the core, the view should first be defined to show only the relevant parts of the model. This will again be the core and coil (the core will be shown as some of the dimensions position the coil relative to the core). The first thing is to set the display for the Coil part. Choose the following menu option:

DISPLAY ↓
Options

Complete the dialogue box and then **Accept**

Refresh the viewing screen

Display border

X minimum

X maximum

Y minimum

Y maximum

Show nodes:

☒ NONE ☐ ALL ☐ REFE ☐ POLY

Show variable names:

☒ NONE ☐ ALL ☐ SAME

+ or -

Show polygon:

☒ NONE ☐ ALL ☐ SAME

+ or -

Choose **Return** to refresh the display and return to the top level menu options. The **Coil** part is created and key variables added to it by completing and following the menu options and **Accept** each dialogue box.

DEM ↓

Create or modify PARTS → Create new part

Part name =

Add part command → Set text

Text message =

Select **Ask for variable** and select **Coil_t** from the list of variables.

Repeat the menu option **Ask for variable** for the following variables:

Coil_h
Coil_ir
Coil_bh

Set the following text:

Select **Set text**

Text message =	Coil properties
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

and then add the following variables to the part with the **Ask for variable** option:

Cur_tur
Num_tur
Wire_d
Wir_rho

Select **Return**, **Quit** and **Return** to return to the top menu level. The display options can be changed first to display show the nodes and their labels using the

DISPLAY ↓
Options

menu item. As with the core, dimension arrows can be added for the 4 dimensions of the coil.

For the variable *Coil_t*, this is done by first constraining two reference points:

DEM ↓
Dimension lines → **By keyboard** → **Vector**

Complete the following dialogue box and then **Accept**

First constrained point =	2.04
Second constrained point =	0.020
X vector =	0
Y vector =	core_g
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

DEM ↓
Dimension lines → **By keyboard** → **Vector**

Complete the following dialogue box and then **Accept**

First constrained point =	0.020
Second constrained point =	0.021
X vector =	coil_t
Y vector =	0
<div style="display: flex; justify-content: space-around;"> <div>Accept</div> <div>Dismiss</div> </div>	

DEM ↓

Dimension lines → **Define/edit dimension**

Select **COIL_T** from the list of variables followed by **Set dimension options**

Complete the following dialogue box and then **Accept**

Create or modify variable	
Variable Descriptor	Radial thickness
Dimension label type:	
<input type="checkbox"/> NONE <input checked="" type="checkbox"/> LINE <input type="checkbox"/> ARC <input type="checkbox"/> CEN <input type="checkbox"/> CENA <input type="checkbox"/> CENC	
Dimension start point	0.020
Dimension end point	0.021
Dimension centre point	
Displacement of line	
Expression for curvature	0
X-shift for the label	0
Y-shift for the label	core_g/2
<div style="display: flex; justify-content: space-around;"> <div>Accept</div> <div>Quit</div> </div>	

Choose the following menu option to set the line arrow styles:

Select **set line style**

Complete the following dialogue box and then **Accept**

Set the line parameters

Dimension line type	0
Line type of end bar	1
Arrow direction:	
<input type="checkbox"/> NONE <input checked="" type="checkbox"/> BOTH <input type="checkbox"/> FORW <input type="checkbox"/> BACK	
Arrow length expression	core_g/4
End bar length expression	core_g*1.5
End bar shift expression	core_g*0.25
<div style="display: flex; justify-content: space-around; width: 100%;"> <div style="border: 1px solid black; padding: 5px 20px;">Update</div> <div style="border: 1px solid black; padding: 5px 20px;">Accept</div> <div style="border: 1px solid black; padding: 5px 20px;">Quit</div> </div>	

Choose **Return** twice to get back to the main modelling menus.

Dimension arrows can be set for the remaining variables if required using the same process

The Target

The final part to be configured is that of the target. Firstly the display parameter will be set up and then the part configured. Hence select the display options as follows:

DISPLAY ↓
Options

Complete the dialogue box and then **Accept**

Refresh the viewing screen

Display border

X minimum

X maximum

Y minimum

Y maximum

Show nodes:

☒ NONE ☐ ALL ☐ REFE ☐ POLY

Show variable names:

☒ NONE ☐ ALL ☐ SAME

+ or -

Show polygon:

☒ NONE ☐ ALL ☐ SAME

+ or -

Ensure labels are hidden by **Hide labels** from the following toggle:

DISPLAY ↓

Hide labels (*Label polygons toggle*)

Choose **Refresh** to refresh the display and return to the top level menu options. The *Target* part is created and key variables added to it by completing and following the menu options and selecting **Accept** after each dialogue box.

DEM ↓

Create or modify a part → **Create new part**

Part name = **Target**

Add part command → **Set text**

Text message = **Target position**

Select **Ask for variable** and then choose **T_to_C** from the list of variables.

Add part command → Set text

Text message =	Target dimensions
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Select **Ask for variable** and then choose **Targ_r** from the list of variables.

Repeat the menu option **Ask for variable** with variable **Targ_h**.

Finally select:

Add part command → Set text

Text message =	Target properties
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Select **Ask for variable** and then choose **Targ_c** from the list of variables.
Select **Show material BH data** and then complete the dialogue box and **Accept**

Material name	FERRITE
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Select **Return** and then **Quit** to close the menuing required to define the part data and then **Return** to return to the top menu bar.

Dimension arrows can be set for the target variables if required.

This now completes the setting up of the Design Environment Module and it is sensible to save the model data. Hence select:

FILE ↓
 Save DEM file

Select **YES** at the confirmation prompt to overwrite the existing model.

Model Testing

The model has now been completed. Often it will be necessary to check that the model is suitable for the problem, and that analysis and post processing commands behave as expected.

There are two main routes for checking the model data. The first is to use the

MODEL ↓

Check OPERA-2d data

menu route. This command converts the parametric model data into a standard OPERA-2d data format. Within this command, there are various facilities available to check on the model mesh quality, print region data and view the model. The option of writing a standard OPERA-2d data file (*.op2) is also available. Such a file can then be solved directly using the relevant analysis module, or read into the pre and post processor.

The other facility for checking the model data is correct is to run the options available in the *User System*. These facilities are replicated in the *Configuration System* for testing purposes under:

ANALYSIS ↓

The user is able to set up an analysis run using the same instructions as described in the User System guide to this example (see [“Induction Sensor User System” on page 3-1](#)). Results from the run should be checked for errors or inconsistencies. For complex post processing scripts, it is advisable to check that the .lp file created during the OPERA-2d post processing stage contains no errors.

If possible a range of models should be run to test the robustness of the module for a wide variety of different dimensions.

Chapter 5

Switched Reluctance Motor User System

The User System

Figure 5.1 shows a typical design for a switched reluctance motor. A parameterised model of this device has been generated using the *Design Environment Configuration System* and is stored in a file called *srm.dem*. This file can also be found in the OPERA-2d installation sub-folder *Examples/2D*. Details of building this model are given in “[Switched Reluctance Motor Configuration](#)” on page 6-1.

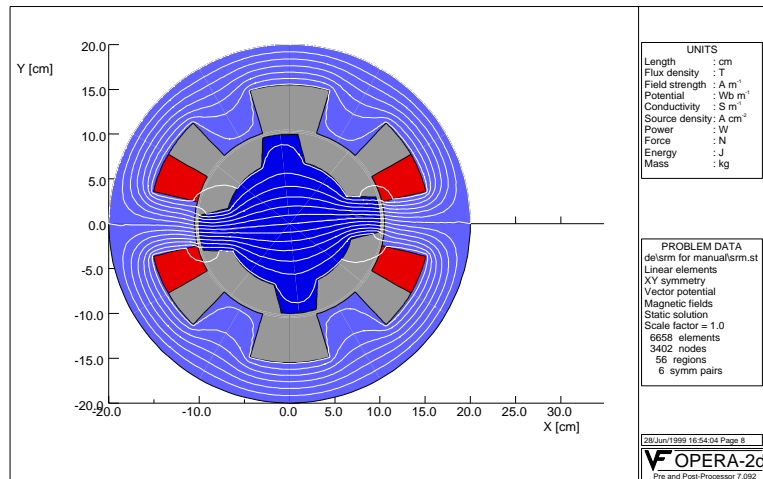


Figure 5.1 A typical switched reluctance motor with flux paths

The figure shown demonstrates a cross-sectional view of a motor. There are 3 main design areas of interest: the coils, stator and rotor. The coils in the stator pro-

vide the appropriate excitation pattern. In this motor, the coils are configured to generate a 2-pole field pattern within the stator.

The stator is generally a non-conducting ferrous material such as steel, which is built up out of laminations to reduce eddy current losses. The rotor is similarly composed of laminations of a non-conducting ferrous type.

Periodic symmetry exists within the geometric and electromagnetic fields generated, only one half of the motor is modelled to help reduce computation times.

This tutorial will be used to calculate the change in generated torque as the rotor is moved through several angles.

Getting Started

A set of models will be created by the use of the *Design Environment* to model variations in the dimensions of the switched reluctance motor. Before starting a new directory, *srm_run1*, will be created which will hold all the files for this analysis using the *srm* Design Environment Module. This will help to keep the files together for easier access later, should any particular file need to be used again.

UNIX Systems

The new directory can be created using

```
> mkdir srm_run1
> cd srm_run1
```

Windows

The start folder can be changed using the OPERA Console Window, selecting

```
OPERA-2d ↓
  Design Environment → Change Project Folder
```

and typing the **name and path** of the new folder, *srm_run1*, into the input box.

All systems

Start the *Design Environment User System*.

To load the Design Environment Module for the switched reluctance motor, select

```
FILE ↓
  Load DEM file
```

Change the directory within the file list box to access the *work/examples/2d* sub-directory of the installation directory, and then select the file *srn.dem*

Running analyses

The Design Environment has the facility for generating multiple variations of a device and analysing them off-line. This is done through the generation of a *Control Set*. This control set contains a list of the variations that are to be generated, including the variable values for each variation and an associated name.

The control set name must be given first. Select

ANALYSIS ↓

Set control name

Filename =	run1
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Two files are created based around this name.

- The control set command script: *run1.comi*

This file contains the list of commands that can be executed by the pre and post processor. These commands access other command scripts that are generated for each model variation that is created within the control set.

- The control set results file: *run1.csr*

This file is a standard file to which RESULTS are written by the post processing command scripts.

Generating variations

The menus available to configure variations of the motor are found under

ANALYSIS ↓

Set PART data

For this analysis run we will change the size of the air gap to a value of 0.05 and rotate the rotor through a range of 5 different angles. To change the values of these rotor parameters, select

ANALYSIS ↓

Set PART data → Rotor

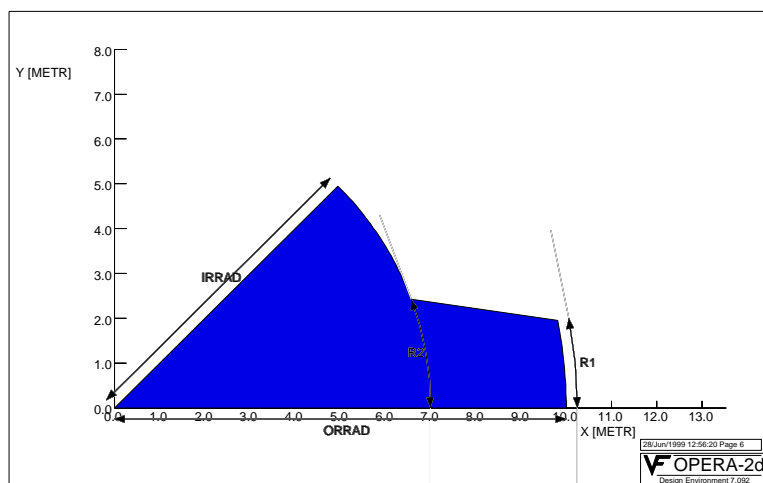


Figure 5.2 Dimensions for the rotor

Core Dimensions			
Outer radius	<input type="text" value="10"/>		
Inner radius	<input type="text" value="7"/>		
Outer % of pole angle	<input type="text" value="25"/>		
Inner % of pole angle	<input type="text" value="45"/>		
Number of rotor poles	<input type="text" value="4"/>		
Rotation angle	<input type="text" value="*"/>		
Airgap thickness	<input type="text" value="0.05"/>		
Rotor BH characteristics			
IRON (Isotropic):	<input type="text" value="default.bh"/>		
Variation filename	<input type="text" value="rotor05"/>		
Start	<input type="text" value="0"/>	Finish	<input type="text" value="10"/>
		Increment	<input type="text" value="2.5"/>
<input type="button" value="Store"/>		<input type="button" value="Update"/>	<input type="button" value="Quit"/>

The use of the * for the 'Rotation angle' parameter means that this variable is to use a range of values, from **Start** to **Finish**, increasing in value by **Increment**.

Select the **Store** button. This creates files containing the model variation data. As we have a range of variable values for the rotation angle of the rotor, a range of model variation files are created. Each model variation filename is generated by appending a number to the filename stem. Therefore the five files, *rotor05_1.var* to *rotor05_5.var*, are created.

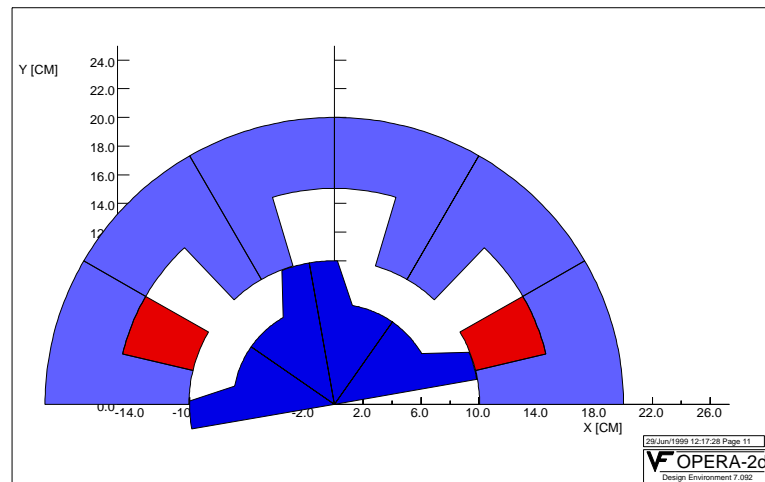


Figure 5.3 Rotor at an angle of 10°

Preparing and Analysing Models

We have created five model variations. All of these can be converted into OPERA-2d data files, analysed and post processed by selecting

ANALYSIS ↓

Prepare and run analysis

This menu option:

- Loads each model variation, and prepares it for analysis and post processing. In doing this an OPERA-2d data file (extension *.op2*) is created, together with a post processing command script file (extension *.script*) containing the commands necessary for solving and post processing this particular file.
- Prepares the Control set command script file to access each of the model variations that is included

- Starts the OPERA-2d pre and post processor¹, and begins the analysis and post processing of all the models.

This command may take some time to execute, and generally the user is advised to run this command at a time when the analysis of the models will not adversely affect users of the computer system, for example it could be started to run over-night.

An alternative to this option is to select **Prepare control set** from the menu. This will create all necessary files to run the analysis. The analysis can then be started by selecting the **Run analysis**, or the pre and post processor can be started and the control set command script run by selecting the file *run1.comi* through

```
FILE↓
    Load a command file
```

Studying results

After analysis, any values stored using the **RESULTS** post processing command, can be viewed. In this case only two values were stored, the rotor angle and calculated torque, To view the results:

```
ANALYSIS ↓
    View analysis results
```

and the file *run1.csr* selected from the file list box. The results from all analyses are listed in a message box.

Individual results are also stored with each model variation. Any of the model variations can be loaded, setting the values of variables to those used for this variation. This is done by selecting:

```
ANALYSIS ↓
    Load model variation
```

and the name of the file selected. For example selecting *airgap05_3.var* will load the model variation with the rotor at an angle of 5 degrees. The results for this particular analysis are displayed when loading the model.

1. Note for UNIX systems the launching of the pre and post processor requires that the alias OPERAPP be defined within the UNIX shell.

Chapter 6

Switched Reluctance Motor Configuration

Switched Reluctance Motor (SRM)

Figure 6.1 shows a typical design for a switched reluctance motor.

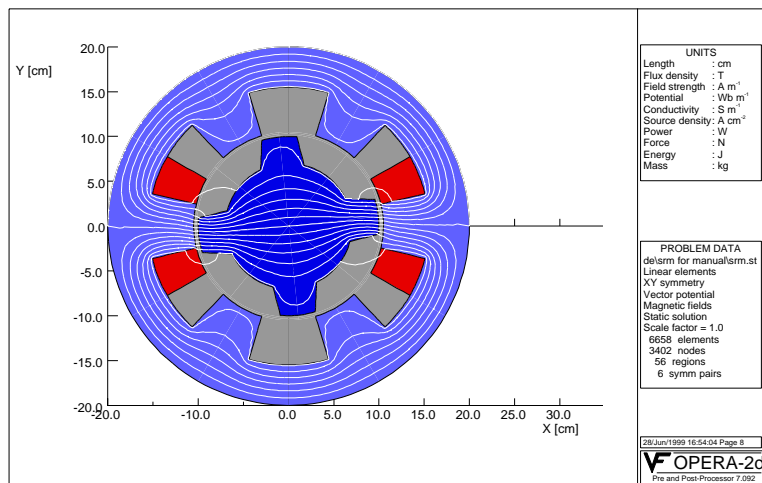


Figure 6.1 A Typical Switched Reluctance Motor

In this example a model of the 2D cross-section of the motor will be generated. This tutorial uses the *Configuration System* as a means of generating a parametric model on a modular basis. The chapter “[Switched Reluctance Motor User System](#)” on page 5-1 shows how this is then used by the user.

Three separate module files will be used.

- *rotor.dem* will contain the model information for the rotor and the air regions in the gap between rotor and stator.
- *stator.dem* will contain model data for the stator and the windings geometries.
- *base.dem* will contain the units, post processing, periodicity conditions and analysis data that is common to both of these particular sections.

These three module files will be combined to form the single module, *motor.dem*, and this will be used to run the analysis of the full switched reluctance device. This modular approach allows replacement of part of the model without needing to change the other sections at all.

Launch the *Configuration System*.

The rotor and airgap

Dimension Scheme

Now the design of the rotor must be considered. Figure 6.2 shows the dimension scheme that will be applied to the rotor.

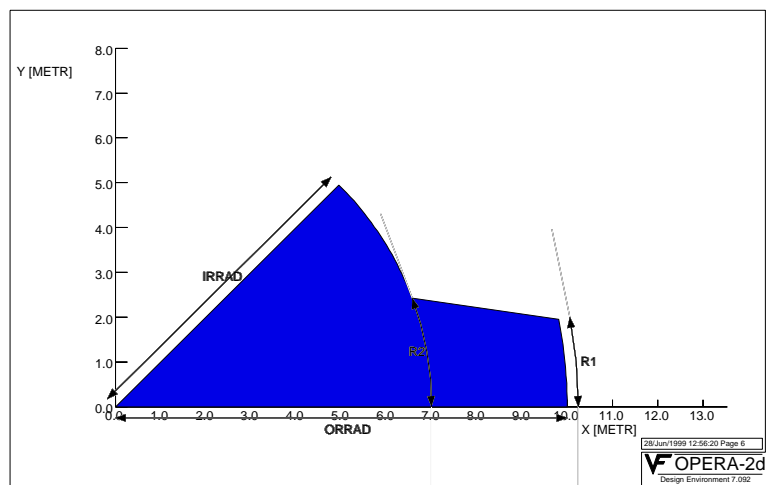


Figure 6.2 Dimension scheme for the rotor

DEM Summary

The data to be defined in this module includes

- User variables required for the rotor design
- Constraints required to define the rotor geometry
- Polygon side data such as curvatures and boundary conditions
- Material properties for the rotor
- Replication parameters to define the whole rotor
- Polygons to model the air regions
- **PART** data to provide a menu for configuring rotor dimensions within the *User System*

Design Variables

The variables that will be used in constraining the rotor geometry will now be defined. This is done using the **VARIABLE** command. In each case the variable name will be given, followed by a value to which the variable is to be set and a description of the variable for later use in defining **PARTS** for the *User System*.

The first variable to be set is the internal radius of the rotor, **IRRad**.

MODEL ↓

Variables → Define new variable

Complete and **Accept** the following dialogue box:

Variable name =	IRRad
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Expression =	7
Descriptor =	Inner radius
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

The next variable to be set is for the outer rotor radius, **OrRad**.

MODEL ↓

Variables → Define new variable → Orrad

complete and **Accept** the following dialogue boxes:

Variable name =	OrRad
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Expression =	10
Descriptor =	Outer radius
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

This menu above is required to define the remaining variables used to define the rotor. The variables to be defined are listed below:

Variable name	Expression	Descriptor
R_POLES	4	Number of rotor poles
R1	25	Outer % of pole angle
R2	45	Inner % of pole angle

Variable name	Expression	Descriptor (<i>continued</i>)
R_ALPHA	$360/(R_poles*2)$	Half pole angle
RANGLE	0	Rotation angle
AIRGAP	0.5	Airgap length

To list the variables that have been defined use:

MODEL ↓

Variables → List variables

A **LIMIT** can be applied to ensure that the outer radius is larger than the inner radius. Select

MODEL ↓

Variables → Limit

variable values → Limit variable expression

Complete the dialogue box as shown and then **Accept**:

Limit Variable Expression

Expression 1

☐ LT
☐ LE
☒ GT
☐ GE
☐ EQ
☐ NE

Expression 2

Associated message

Other limits can also be applied using this menu. Such limits could e.g. keep the pole angles within the range 0 to 100% or restrict the maximum and minimum number of poles. The module to be produced in this section requires an even number of rotor poles. This can be enforced using a limit on the value of **R_Poles**. Select

MODEL ↓

Variables → Limit

variable values → Limit variable expression

Limit Variable Expression	
Expression 1	<input type="text" value="Mod(R_poles;2)"/>
<input type="checkbox"/> LT <input type="checkbox"/> LE <input type="checkbox"/> GT <input type="checkbox"/> GE <input checked="" type="checkbox"/> EQ <input type="checkbox"/> NE	
Expression 2	<input type="text" value="0"/>
Associated message	<input type="text" value="Invalid number of poles"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

The **MOD** function returns the remainder when the first argument is divided by the second. So in this expression the value must be exactly divisible by two.

Rotor geometry and constraints

A polygon must now be defined to model the rotor. A 5-sided polygon will be used to model one half of one pole of the rotor. Replications of this polygon will subsequently be used to model the complete rotor.

MODEL ↓

Geometry → New polygon

Complete and **Accept** the following dialogue box:

Number of polygon sides =	<input type="text" value="5"/>
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

A pentagon will appear centred on the screen. It is necessary to constrain this polygon to fit the dimension scheme shown earlier.

The first point in the polygon will be constrained to lie at the origin. This is done using the **POINT** constraint by entering

MODEL ↓

Constraints → By keyboard → Fixed point

Complete the dialogue box as shown and then **Accept**:

Point name =	1.01
X position =	0
Y position =	0
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

and the geometry will change to position point 1.01 at (0,0). The point 1.02 will now be constrained at a fixed radius length from point 1.01.

MODEL ↓

Constraints → By keyboard → Length

First constrained point =	1.01
Second constrained point =	1.02
Length =	OrRad
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

The point 1.02 has been constrained to be the correct distance from point 1.01, but a further constraint is required to set the angle adopted by point 1.02 with respect to point 1.01.

MODEL ↓

Constraints → By keyboard → Angle

First constrained point =	1.01
Second constrained point =	1.02
Angle =	RAngle
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Point 1.03 will be constrained by adopting an angle set by the variable **R1** as a percentage of the half-pole angle, and by fixing its distance to the same radius as point 1.02. Select

MODEL ↓

Constraints → By keyboard → Internal Angle

First constrained point =	1.03
Second constrained point =	1.01
Third constrained point =	1.02
Angle =	$R_Alpha * R1 / 100$
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

The next constraint sets the difference in length between points 1.01 to 1.03 and points 1.01 to 1.02 to be zero, i.e. they are the same length¹. Select

MODEL ↓

Constraints → By keyboard → Length Difference

Complete the following dialogue box as shown and then **Accept**:

First point of first length =	1.01
Second point of first length =	1.03
First point of second length =	1.01
Second point of second length =	1.02
Length =	0
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Points 1.04 and 1.05 will be constrained using the mouse cursor mode. This is done as follows:

Choose **Return** to step back a menu and then select

MODEL ↓

Constraints → By cursor

A message box will appear stating that **POINT** constraints are currently being added. Select **Continue** to clear the message and then select **Constraint type** to select a different type of constraint. Select **Internal angle** and after clearing the message, select **Select points**. Press **F1** to temporarily remove the menus. Click the cursor at (10,8), (0,0) and (10,0) to select points 1.04, 1.01 and 1.02, in that

1. A length constraint between points 1.01 and 1.03 set at **OrRad** would have the same effect as this constraint.

order. Once these points have been chosen, a dialogue box will appear requiring the expression of the angle. Complete the parameter box and then **Accept**

Expression =	R_Alpha*R2/100
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Point 1.04 now needs to be constrained to lie a certain distance from the origin. Hence the constraint type required is a length constraint. Press **F1** to retrieve the menus, select **Constraint type** and then **Length** from the menu list. After clearing the message box, select **Select points**. Press **F1**, click the mouse cursor at (0,0) and (10,8) to select points 1.01 and 1.04 and then complete and **Accept** the parameter box as follows:

Expression =	Irrad
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Press **F1** to retrieve the menu and select the **Repeat last constraint** options. Press **F1** to clear the menu and click the mouse cursor at (0,0) and (4,6) to select points 1.01 and 1.05 so as to constrain the distance between them. As the constraint is being repeated, the previous expression will be used. Hence once the two points have been selected, the constraint will be added and a message will appear confirming this.

Point 1.05 requires one final constraint to define its location. Hence retrieve the menu through pressing **F1** and select **Constraint type** from the menu list. Choose **Angle** and clear the message. Select **Select points**, press **F1** and click the mouse cursor at (0,0 and (4,6) to select points 1.01 and 1.05. Complete the following parameter box and **Accept**.

Expression =	R_Alpha+Rangle
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

All the points of the rotor polygon have now been constrained. Press **F1** and select the menu option **Solve** to solve the constraints and generate the correct topology.

At present all sides of the polygon are straight. The next step is to define some curvatures for the polygon sides. Select **Return** twice and then

MODEL ↓

Geometry → Set side curvature

Complete and **Accept** the following dialogue box:

Set side curvature:		
Curvature type:		
<input type="checkbox"/> STRAight	<input type="checkbox"/> CURVed	<input checked="" type="checkbox"/> CENTred
Centred on point	<input type="text" value="1.01"/>	
Curvature value	<input type="text"/>	
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>	

Select the **By cursor** option and then press **F1** to remove the menu options, which are obscuring the polygon. Click the mouse cursor at about point (10,1) between points 1.02 and 1.03 and click at about point (6,3) between points 1.04 and 1.05. Press **F1** to retrieve the menu options and select **Return**. The curvatures have been applied. The next step is to define the material properties of the rotor. Select **Return** to close down the menus.

Airgap geometry and constraints

The airgap will also be modelled as part of the rotor. Three polygons need to be generated to model the airgap.

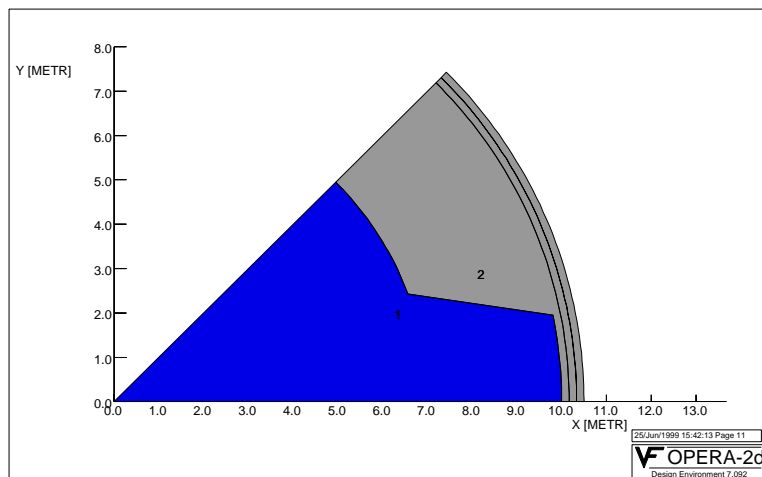


Figure 6.3 The 4 regions used to model the rotor and airgap

The three polygons are needed to ensure that the airgap is well modelled to produce accurate calculations for torque. To create the first airgap polygon select:

MODEL ↓
Geometry → **New polygon**

Fill in and accept the following dialogue box

Number of polygon sides =	6
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

A hexagon will appear centred on the screen. This needs to be constrained. However, 3 of the faces of this polygon will be common with faces from the previously defined rotor. These faces can be linked together using

MODEL ↓
Constraints → **Link sides** → **By keyboard**

Master side =	1.04
Slave side =	2.01
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

This moves side 1 (the side after point 1) of polygon 2 so that it is matching side 4 (the side after point 4) of polygon 1. Two further links can be made. Select

MODEL ↓
Constraints → **Link sides** → **By keyboard**

Master side =	1.03
Slave side =	2.02
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

MODEL ↓
Constraints → **Link sides** → **By keyboard**

Master side =	1.02
Slave side =	2.03
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The three sides of the rotor are now matched by the 3 sides of the air region. This leaves only two points in the air region that are unconstrained. These will be constrained so that the region fills 1/3 of the airgap. The other two regions to be added will complete the airgap. In this way there are three layers of regions, and hence at least three layers of elements through the airgap. This is necessary to ensure accurate post processing results can be obtained for the torque.

The two points will be constrained to be at a distance of $\text{airgap}/3$ beyond the outside of the rotor, and at the correct angles. In this case length difference, angle difference and internal angle constraints will be used. Select

MODEL ↓

Constraints → By keyboard → Length Difference

First point of first length =	1.01
Second point of first length =	2.05
First point of second length =	1.01
Second point of second length =	2.04
Length =	Airgap/3
<div>Accept</div> <div>Dismiss</div>	

MODEL ↓

Constraints → By keyboard → Angle difference

First point of first angle =	1.01
Second point of first angle =	2.05
First point of second angle =	1.01
Second point of second angle =	2.04
Angle =	0
<div>Accept</div> <div>Dismiss</div>	

MODEL ↓

Constraints → By keyboard → Length Difference

First point of first length =	1.01
Second point of first length =	2.06
First point of second length =	1.01
Second point of second length =	2.05
Length =	0
<div>Accept</div> <div>Dismiss</div>	

MODEL ↓

Constraints → By keyboard → Internal angle

First constrained point =	2.06
Second constrained point =	1.01
Third constrained point =	2.05
Angle =	R_Alpha
<div>Accept</div> <div>Dismiss</div>	

This completes the constraints for the first polygon in the air region. The next polygon will be added and constrained.

MODEL ↓

Geometry → New polygon

Number of polygon sides =	4
<div>Accept</div> <div>Dismiss</div>	

A quadrilateral will appear centred on the screen. One face of this polygon will be common with a face from the first polygon used for the air region. These faces can be linked together using

MODEL ↓

Constraints → Link sides → By keyboard

Master side =	2.05
Slave side =	3.01
<div>Accept</div> <div>Dismiss</div>	

Constraints for the two points of this polygon that are unconstrained can be added.

Select

MODEL ↓

Constraints → By keyboard → Length Difference

First point of first length =	1.01
Second point of first length =	3.03
First point of second length =	1.01
Second point of second length =	3.02
Length =	Airgap/3

Accept

Dismiss

MODEL ↓

Constraints → By keyboard → Angle difference

First point of first angle =	1.01
Second point of first angle =	3.03
First point of second angle =	1.01
Second point of second angle =	3.02
Angle =	0

Accept

Dismiss

MODEL ↓

Constraints → By keyboard → Length Difference

First point of first length =	1.01
Second point of first length =	3.04
First point of second length =	1.01
Second point of second length =	3.03
Length =	0

Accept

Dismiss

MODEL ↓

Constraints → By keyboard → Internal angle

First constrained point =	3.04
Second constrained point =	1.01
Third constrained point =	3.03
Angle =	R_Alpha
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

This completes the constraints for the third polygon. The final polygon will be added and constrained.

MODEL ↓

Geometry → New polygon

Number of polygon sides =	4
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

As before, a single face can be linked to the previous region.

MODEL ↓

Constraints → Link sides → By keyboard

Master side =	3.03
Slave side =	4.01
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

and the remaining two points will be constrained by

MODEL ↓

Constraints → By keyboard → Length Difference

First point of first length =	1.01
Second point of first length =	4.03
First point of second length =	1.01
Second point of second length =	4.02
Length =	Airgap/3
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

MODEL ↓

Constraints → By keyboard → Angle difference

First point of first angle =	1.01
Second point of first angle =	4.03
First point of second angle =	1.01
Second point of second angle =	4.02
Angle =	0
<div>Accept</div> <div>Dismiss</div>	

MODEL ↓

Constraints → By keyboard → Length Difference

First point of first length =	1.01
Second point of first length =	4.04
First point of second length =	1.01
Second point of second length =	4.03
Length =	0
<div>Accept</div> <div>Dismiss</div>	

MODEL ↓

Constraints → By keyboard → Internal angle

First constrained point =	4.04
Second constrained point =	1.01
Third constrained point =	4.03
Angle =	R_Alpha
<div>Accept</div> <div>Dismiss</div>	

This completes the constraints for the final polygon.

It can be seen that the sides of the polygon are straight. To set the centre of curvature of the sides to be centred on the origin, i.e. point 1.01, select

MODEL ↓

Geometry → Set side curvature

Complete and **Accept** the following dialogue box:

Set side curvature:

Curvature type:

☐ STRAight
 ☐ CURVed
 ☒ CENTred

Centred on point 1.01

Curvature value

Accept
Quit

Select **By keyboard** and complete as shown below

Side = 2.05

Accept
Dismiss

Repeat this for sides **3.03** and **4.03** to complete all of the curvatures for these regions.

Material Properties

The material properties of the region should now be set. The rotor will be non-conducting and will normally be run as a non-linear problem, so permeability need not be parameterized. Therefore, the material properties are set as

- A non-linear material, using the default BH data **IRON**
- Zero Conductivity
- A permeability of 1000 (for linear testing purposes)
- A subdivision size based on the outer radius

This is done by

MODEL ↓

Material data → **By keyboard**

Polygon number = 1

Accept
Dismiss

Complete the following dialogue box and **Accept**

Set the material properties	
<input checked="" type="checkbox"/> Polygon	<input type="checkbox"/> Background
AIR, CONDUCTOR, IRON	IRON
Maximum subdivision	0rrad/10
Permeability	1000
Conductivity	0
Current density	0
Phase of the region	0
Velocity of region	0
Conductor number	0
<input checked="" type="checkbox"/> Connected	<input type="checkbox"/> Disconnected
Include region in OP2 file	1
Accept	Quit

The material properties must be set for the three air regions. This is the default settings, but the subdivision size still needs to be set correctly.

MODEL ↓

Material data → By keyboard

Polygon number =	2
Accept	Dismiss

Set the material properties	
<input checked="" type="checkbox"/> Polygon	<input type="checkbox"/> Background
AIR, CONDUCTOR, IRON	<input type="text" value="AIR"/>
Maximum subdivision	<input type="text" value="0rad/10"/>
Permeability	<input type="text" value="1"/>
Conductivity	<input type="text" value="0"/>
Current density	<input type="text" value="0"/>
Phase of the region	<input type="text" value="0"/>
Velocity of region	<input type="text" value="0"/>
Conductor number	<input type="text" value="0"/>
<input checked="" type="checkbox"/> Connected	<input type="checkbox"/> Disconnected
Include region in OP2 file	<input type="text" value="1"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

Repeat this for polygons **3** and **4** as well

Replicating polygons

Currently only one half of one pole is modelled. Replications will be used to complete the model of the rotor and airgap.

To generate these replications select

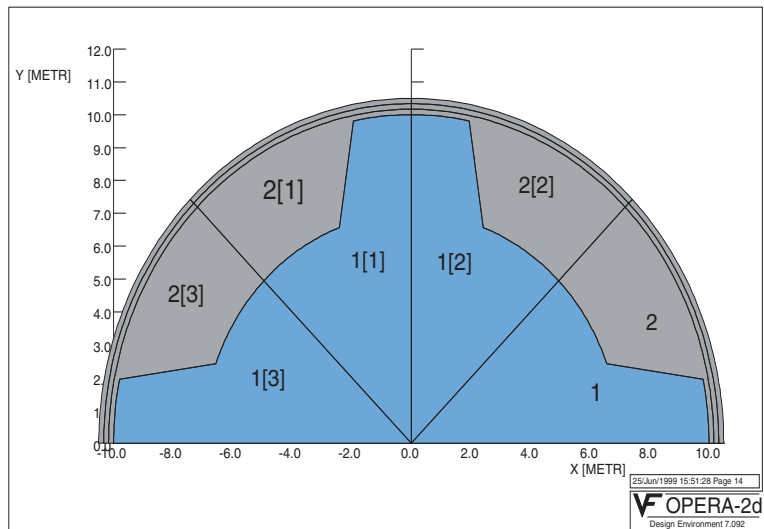


Figure 6.4 The complete rotor and airgap regions with replications

MODEL ↓

Geometry → Replicate polygon → By keyboard

Polygon number = 1

Accept

Dismiss

Set the replication parameters

No. copies x-direction

0

X-dir. copies displacement

0

No. copies y-direction

0

Y-dir. copies displacement

0

☒ reflect on

☐ reflect off

Start line reflect at

1.01

End line reflect at

1.05

No. rotation copies

$(R_poles/2) - 1$

Centre pt of rotation

1.01

Angle of rotation

$2 * R_Alpha$

Accept

Quit

The replication menu above reflects the original polygon in a mirror line defined by the two points 1.01 and 1.05. Having reflected the original polygon to generate an image, these are now rotated about a centre point defined as point 1.01. The polygons are rotated by an angle $2 * R_Alpha, (R_Poles/2) - 1$ times. Any change in the original polygon, such as a material or geometric alteration, is also applied to any replication. The use of these replications is necessary to allow the change in the number of poles in the rotor.

Repeat the application of this replication data for the three air polygons 2, 3 and 4.

Boundary conditions

Symmetry boundary conditions need to be applied to the rotor as only half the motor is being modelled, i.e. a periodicity condition of 180° will be applied.

Select:

MODEL ↓
Boundary conditions

Complete the following dialogue box and **Accept**

Set boundary condition

Type of condition:

☐ V
 ☐ DV
 ☐ CLEAR
 ☒ SYMM

Expression for potential value:

Value

Select **By cursor** and use the cursor to select the faces to which the symmetry boundary condition will be applied. Select the face from each polygon on the x-axis, and the outside edge of the outside polygon of the airgap by selecting at the following point:

5, 0

To help select the small sides of the airgap regions on the x-axis, select

Zoom → Zoom in

and using the left mouse button drag a small rubber band box around this region. The next 4 sides can be selected easily using **By cursor**, and selecting at the points:

```
10.1,    0
10.25,   0
10.4,    0
10.4,    0.5
```

Select **Return** from the menu to accept these selections.

Some of these sides will be internal to the model. For these sides, the boundary condition will be removed when converting to the OPERA-2d model for analysis.

Improving the mesh

The model will now be checked to ensure that it produces a good model and mesh, although it will not be possible to analyse it as the model is incomplete.

Select

```
MODEL ↓
    Check OPERA-2d data
```

This will convert the model to the OPERA-2d data form. A list of checks are made. Warnings about unmatched symmetry sides are reported, but can be ignored as the periodicity condition used to match symmetry sides will not be applied until later.

The mesh can be viewed using

```
MODEL ↓
    Check OPERA-2d data → Generate mesh
```

The boundary lines shown here should only be the outside edge, and all sides of it should be marked by crosses to show that symmetry boundary conditions have been applied to all edges as shown in [Figure 6.5, on page 6-23](#).

```
MODEL ↓
    Check
    OPERA-2d data → View pre-
                    processor model → +mesh (-mesh toggle)

    Check
    OPERA-2d data → View pre-
                    processor model → Refresh
```

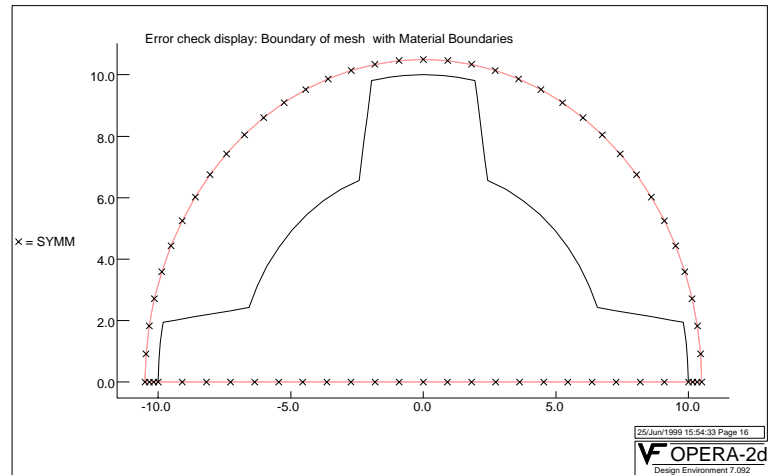



Figure 6.5 Outline of boundaries of the model of the rotor

The airgap will be changed to a smaller, more usual value of 0.05cm. This is done by

MODEL ↓

Variables → **Modify variable**

Select **Airgap** from the list of variables, and enter the new value of **0.05** in the menu box displayed. Check the OPERA-2d data and mesh again.

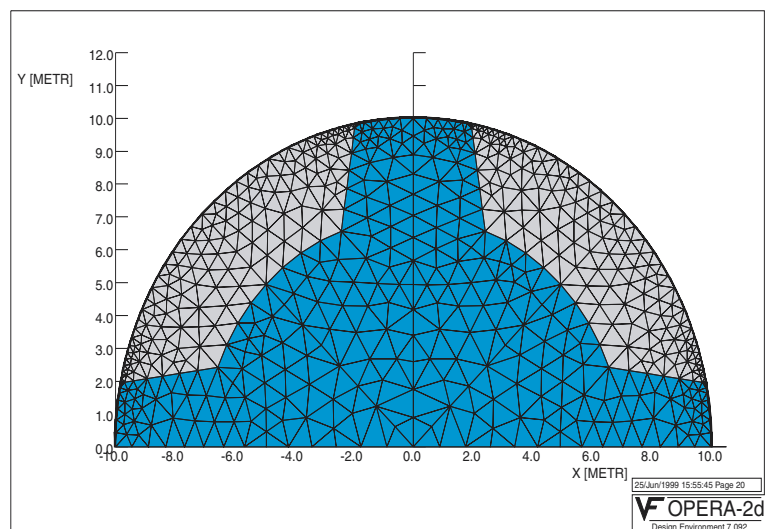


Figure 6.6 The mesh generated with mesh control points

This time it can be noted that the mesh at the corner of the rotor needs to be smaller to model the flux variation here. This can be improved using mesh control points. These points increase the mesh density in a local area of the model. Return to the main MODEL menu and select

MODEL ↓

Geometry → Mesh control point

Set mesh	
Mesh point name	<input type="text" value="1.03"/>
Subdivision at point	<input type="text" value="airgap/3"/>
Radius of influence	<input type="text" value="airgap/3"/>
Associated polygon	<input type="text" value="0"/>
Mesh point options	
<input type="checkbox"/> Delete	<input type="checkbox"/> List <input checked="" type="checkbox"/> None
First point	<input type="text" value="1"/>
Last point	<input type="text" value="*"/>
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

This mesh control point will increase the mesh density near the outside edge of the rotor pole by decreasing the subdivision size as seen in Figure 6.6.

Preparing the Rotor PART for the User System

The rotor is now complete and various parameters can be changed, and the model will vary accordingly. To assist another user of the software a *User System* is available. This has only restricted options and specially prepared menus and views, unique to the model that has been built. These views and menus must be set up in the *Configuration System*. This is done by creating a **PART**.

First, the display will be set to the view that the *User System* will display when variables for the rotor are being set. The size of the screen will be stored as expressions so that changes in dimension can be reflected in the **PART** view. The view will not include the air regions, or any of the labels currently displayed. Select:

DISPLAY ↓

Hide labels (*Label polygons toggle*)

DISPLAY ↓

Options

Refresh the viewing screen	
Display border	
X minimum	<input type="text" value="-orrad"/>
X maximum	<input type="text" value="orrad*0.5"/>
Y minimum	<input type="text" value="0.0"/>
Y maximum	<input type="text" value="orrad*1.5"/>
Show nodes:	
<input checked="" type="checkbox"/> NONE	<input type="checkbox"/> ALL <input type="checkbox"/> REFE <input type="checkbox"/> POLY
Show variable names:	
<input checked="" type="checkbox"/> NONE	<input type="checkbox"/> ALL <input type="checkbox"/> SAME
+ or -	<input type="text"/>
Show polygon:	
<input checked="" type="checkbox"/> NONE	<input type="checkbox"/> ALL <input type="checkbox"/> SAME
+ or -	<input type="text" value="+1"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

DISPLAY ↓

Refresh

The view now only shows the rotor. This view is stored and the part created by

DEM ↓

Create or modify PARTS → Create new part

Complete and Accept the menu box

Part name =	Rotor
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The system is now waiting for different items that should be associated with this **PART**. To set a descriptive text message at the top of the **PART** menu, select:

Add part command → Set text

Complete the dialogue box with the text string and **Accept**

Text message =	Rotor dimensions
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Having set some text, add the variables whose values can be set in this **PART**.

Select **Ask for variable**, and from the list of variables select **Orrad** followed by **Accept**.

Repeat this for the following 5 variables:

Ask for variable → **irrad**
 Ask for variable → **r1**
 Ask for variable → **r2**
 Ask for variable → **R_poles**
 Ask for variable → **Rangle**
 Ask for variable → **Airgap**

The following text is now added to the list of commands associated with the part:

Select **Set text**

Complete the dialogue box with the text string and **Accept**

Text message =	Rotor BH characteristics
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The material **IRON** will need to have a suitable BH curve assigned to it. Hence the user will need to be prompted to give a suitable BH data file:

Select **Show material BH data**

Complete the dialogue box and **Accept**

Material name	IRON
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The PART menu created can be demonstrated and tested by selecting

DEM ↓

Demonstrate a part → **Rotor**

This menu is the means by which the key design variables for the rotor can be easily set. By changing the value of any of the variables, the parametric model can then be updated by selecting the **Update** button.

To close the menu select **Quit**.

The rotor DEM has now been completed and needs to be saved, select

FILE ↓
 Save as new DEM file

Save the DEM file as *rotor.dem*.

Having saved the DEM file, it is necessary to reset the *Configuration System* before starting the stator DEM. This is to remove the variables, constraints and polygons. Select:

OPTIONS ↓
 Clear and reset

and confirm the message box by clicking the mouse cursor on **YES**.

The stator and windings

Dimension Scheme

The design of the stator can be now considered. Figure 6.7 shows the dimension scheme that will be applied to the stator.

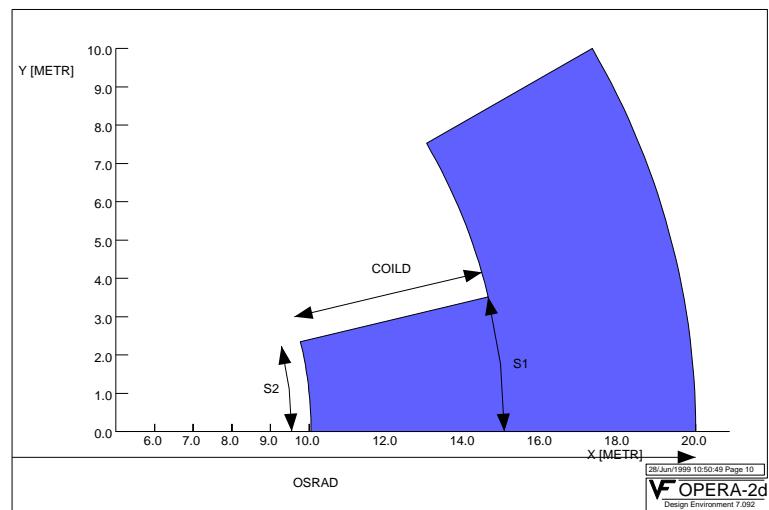


Figure 6.7 Dimension scheme attached to the stator

DEM Summary

The data to be defined in this DEM is listed as follows:

- Design variables required for the stator design
- Constraints required to define the stator and winding geometries
- Polygon side data such as curvatures and boundary conditions
- Material properties for the stator
- Winding characteristics
- Replication parameters to define the whole stator
- PART data to provide menus for the *User System*

Design Variables

The variables that will be used in constraining the stator's geometry must now be defined. Although some of the variables to be used were previously defined in the modelling of the rotor, these variables must be redefined for use within this module.

This is done using the **VARiable** command. In each case the variable name will be given, followed by a value to which the variable is to be set and a description of the variable for later use in the *User System*.

First, set the two variables, **ORRAD** and **AIRGAP**, that are common with those defined in the rotor and are needed for use within the stator as well. The new variables for parameterisation of the stator and the windings will then be defined.

The first variable, **ORRAD**, can be set using

MODEL ↓

Variables → Define new variable

Variable name = **OrRad**

Accept
Dismiss

Expression = **10**
 Descriptor = **Outer radius**

Accept
Dismiss

The other variables to be set using this menu route are

Variable name	Expression	Descriptor
AIRGAP	0.5	Airgap length
S_POLES	6	Number of stator poles
OSRAD	20	Outer radius
ISRAD	OrRad+AirGap	Inner stator radius
S1	45	Inner % of pole angle
S2	45	Outer % of pole angle
S_ALPHA	$360/(S_poles*2)$	Half pole angle
COILD	5	Coil slot depth
CURR	1	Current per turn
NTURNS	200	Number of turns

The variable **OsRad** would benefit from having a value limit applied, to prevent it taking values smaller than is physically possible. Hence a **LIMIT** is required. Select the following menu path:

MODEL ↓

Variables → Limit

variable values → Limit variable expression

Limit Variable Expression	
Expression 1	<input type="text" value="OsRad"/>
<input type="checkbox"/> LT <input type="checkbox"/> LE <input checked="" type="checkbox"/> GT <input type="checkbox"/> GE <input type="checkbox"/> EQ <input type="checkbox"/> NE	
Expression 2	<input type="text" value="IsRad+CoilD"/>
Associated message	<input type="text" value="Outer radius is too small"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

Other limits could also be applied using this menu route. Such limits could e.g. keep the pole angles within the range 0 to 100% or restrict the maximum and minimum number of poles. The module to be produced in this section requires an even number of stator poles. This can be enforced using two limits on the value of **S_Poles** of

MODEL ↓

Variables → Limit

variable values → Limit variable expression

Limit Variable Expression	
Expression 1	<input type="text" value="Mod(S_poles;2)"/>
<input type="checkbox"/> LT <input type="checkbox"/> LE <input type="checkbox"/> GT <input type="checkbox"/> GE <input checked="" type="checkbox"/> EQ <input type="checkbox"/> NE	
Expression 2	<input type="text" value="0"/>
Associated message	<input type="text" value="Invalid number of poles"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

Geometry and constraints

A polygon must now be defined to model the stator. This is done using the **POLYGON** command which takes only a single parameter - the number of points in the polygon. A 6-sided polygon will be used to model one-half of one pole of the stator. Replications of this polygon will be subsequently used to model the complete stator.

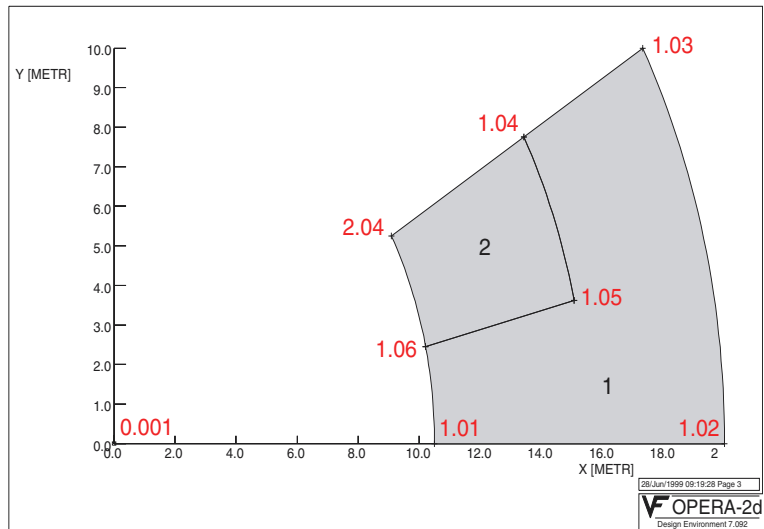


Figure 6.8 The two polygons used to generate the stator and winding regions

MODEL ↓

Geometry → New polygon

Number of polygon sides =	6
Accept	Dismiss

A hexagon appears centred on the screen.

It is now necessary to constrain the geometry to fit the dimension scheme shown earlier, and so choose **Return** to return to the main modelling menu options.

A reference point needs to be placed at the origin. This is achieved by using a **POINT** constraint

MODEL ↓

Constraints → By keyboard → Fixed point

Point name =	0.001
X position =	0
Y position =	0
Accept	Dismiss

This will generate a reference point called **0.001**, and constrain its position to lie at the origin.

The first point in the polygon will be constrained to lie at a point **IsRad** from the origin. This is done using the **LENGTH** and **ANGLE** constraints by entering:

MODEL ↓

Constraints → By keyboard → Length

First constrained point =	0.001
Second constrained point =	1.01
Length =	IsRad
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Point 1.01 now needs to be constrained at a particular angle with respect to the origin. The angle chosen will be the constant, thus fixing the stator orientation, as there is no need for the stator to be able to rotate. Select the menu option

MODEL ↓

Constraints → By keyboard → Angle

First constrained point =	0.001
Second constrained point =	1.01
Angle =	0
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Points 1.02 and 1.03 will be constrained in a similar way.

MODEL ↓

Constraints → By keyboard → Length

First constrained point =	0.001
Second constrained point =	1.02
Length =	OsRad
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

MODEL ↓

Constraints → By keyboard → Angle

First constrained point =	0.001
Second constrained point =	1.02
Angle =	0
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

MODEL ↓

Constraints → By keyboard → Length

First constrained point =	0.001
Second constrained point =	1.03
Length =	OsRad
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

MODEL ↓

Constraints → By keyboard → Angle

First constrained point =	0.001
Second constrained point =	1.03
Angle =	S_Alpha
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Points 1.04, 1.05 and 1.06 will be constrained from commands entered via the cursor mode. Select

MODEL ↓

Constraints → By cursor → Constraint type → Length

Constraints → By cursor → Select points

Using the cursor select points 0.001 and 1.04 at:

```
0,      0
12,     7
```

and select **Accept** after completing the dialogue box:

Expression =	IsRad+CoilD
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

The same constraint can be repeated for point 1.05. Select

Repeat last constraint

and select points 0.001 and 1.05 by

```
0,      0
8,      8
```

A message box will appear confirming the constraint addition.

The display can be resized.

Select

MODEL ↓

Constraints → By cursor → Zoom → Bounding box

Constraints → By cursor → Constraint type → Angle

and after clearing the message box, select

MODEL ↓

Constraints → By cursor → Select points

Select points 0.001 followed by 1.04 at:

0, 0

12, 7

and complete the dialogue box:

Expression =	S_Alpha
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

For constraining the angle of point 1.05, select points 0.001 and 1.05 by:

MODEL ↓

Constraints → By cursor → Select points

and select the points at:

0, 0

8, 8

Expression =	S_Alpha*s1/100
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The effect of these constraints can be seen by solving for the new constraints that have been added. Select

MODEL ↓

Constraints → By cursor → Solve

Finally point 1.06 will be constrained using

MODEL ↓

Constraints → By keyboard → Angle

First constrained point =	0.001
Second constrained point =	1.06
Angle =	S_Alpha*s2/100
<div>Accept</div> <div>Dismiss</div>	

MODEL ↓

Constraints → By keyboard → Length

First constrained point =	0.001
Second constrained point =	1.06
Length =	IsRad
<div>Accept</div> <div>Dismiss</div>	

The polygon is now fully constrained.

Only length and angle constraints have been used above. Alternative constraint schemes could also have been used and would have given the same solution. In this case, the use of **LENGTH** and **ANGLE** constraints from the single fixed point, **0.001**, will always guarantee the required unique solution to the constraint set.

A 4-sided polygon will be created to represent the windings within the model.

MODEL ↓

Geometry → New polygon

Number of polygon sides =	4
<div>Accept</div> <div>Dismiss</div>	

To constrain this polygon, link two of the sides to the sides of the polygon defining the stator. This will constrain the position of three of the polygon corners. The fourth point will be constrained using length and angle constraints.

MODEL ↓

Constraints → Link sides → By keyboard

Master side =	1.05
Slave side =	2.01
<div>Accept</div> <div>Dismiss</div>	

This has linked the first side of polygon 2 will be positioned against the inside edge of the stator pole.

MODEL ↓

Constraints → Link sides → By keyboard

Master side =	1.04
Slave side =	2.02
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

This links the second side of the polygon to the inner edge of the stator.

To constrain the final point of this polygon, select

MODEL ↓

Constraints → By keyboard → Angle

First constrained point =	0.001
Second constrained point =	2.04
Angle =	S_Alpha
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

MODEL ↓

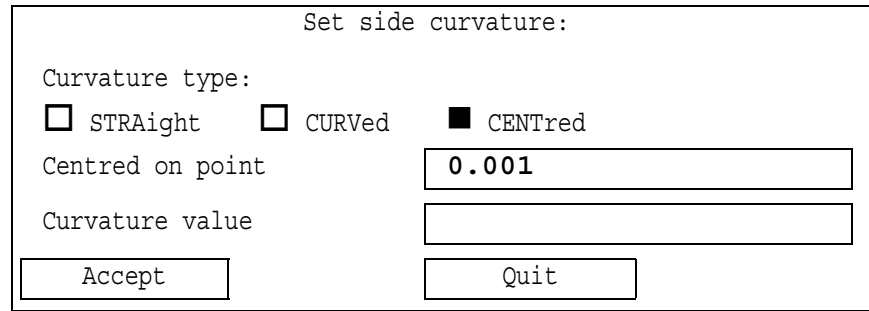
Constraints → By keyboard → Length

First constrained point =	0.001
Second constrained point =	2.04
Length =	IsRad
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The sides of these polygons must now be set to the correct curvature. This is done by

MODEL ↓

Geometry → Set side curvature



Set side curvature:

Curvature type:

☐ STRAight ☐ CURVed ☒ CENTred

Centred on point

Curvature value

Select **By cursor** to select the 4 sides that are to be curved. Press the **F1** key to hide the menu options, which are obscuring the display and click the mouse cursor at

10.5, 1
10, 4
19, 5
15, 5

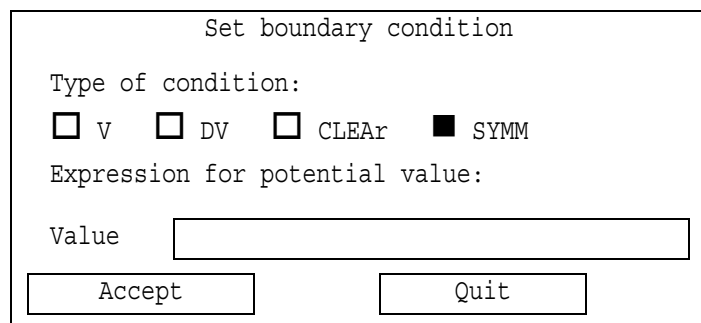
Press the **F1** key to show the menus again, followed by **Return** to accept this selection. The inside edge of the stator, inside edge of the winding, outside edge of the winding / inside edge of the stator, and the outside edge of the stator all now have a curvature centred on the origin.

Boundary conditions

Symmetry boundary conditions need to be applied to the stator as a half model is used. The applied boundary conditions will be ignored if they lie on an internal region boundary when converting to OPERA-2d data files.

MODEL ↓

Boundary conditions



Set boundary condition

Type of condition:

☐ V ☐ DV ☐ CLEAR ☒ SYMM

Expression for potential value:

Value

Select **By cursor** and use the cursor to select the three faces to which the symmetry boundary condition will be applied. Select the face from the stator polygon on the x-axis, the inside edge of the stator and the inside edge of the winding polygon by

```
15.0, 0
10.5, 1
10, 4
```

and select **Return** from the menu to accept these selections.

Another boundary condition is required to set the potential on the exterior of the stator to zero. Select

MODEL ↓

Boundary conditions

Clear the message box confirming the boundary condition to be applied and then select **By cursor** and select the outside edge of the stator, side 1.02, by clicking at

```
19, 5
```

followed by **Return** to accept this selection.

This will apply the potential ($V=0$) boundary condition on the side located between points 1.02 and 1.03. This boundary condition is equivalent to forcing a tangential field, which implies that no flux leaks out of the back of the stator.

Replicating and copying polygons

The polygons now need to be replicated to allow the definition of the full geometry. The winding will be copied rather than replicated. This is because the sign and magnitude of the current density will vary in different sections of the winding.

Select

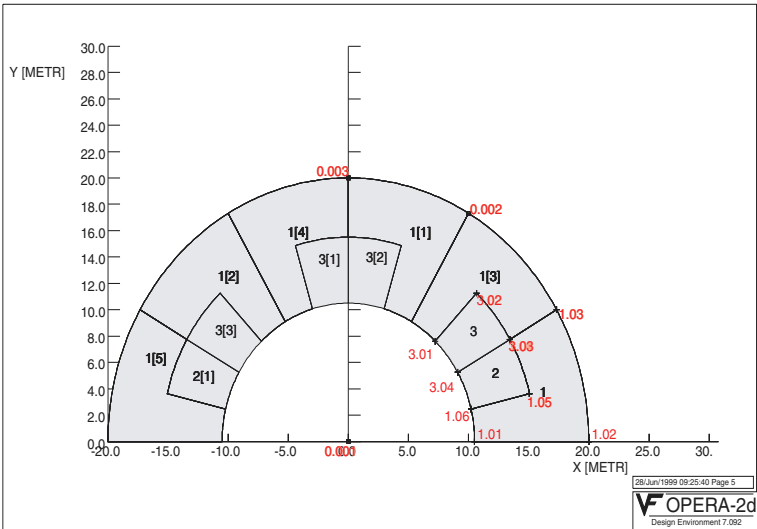


Figure 6.9 Complete stator model with copies and replications

MODEL ↓

Geometry → Replicate polygon → By keyboard

Polygon number = 1

AcceptDismiss

Set the replication parameters

No. copies x-direction	0
X-dir. copies displacement	0
No. copies y-direction	0
Y-dir. copies displacement	0
<input checked="" type="checkbox"/> reflect on	<input type="checkbox"/> reflect off
Start line reflect at	0.001
End line reflect at	1.03
No. rotation copies	(S_poles/2) - 1
Centre pt of rotation	0.001
Angle of rotation	2*S_Alpha
Accept	Quit

This generates the complete geometry for the stator. The replications of this region lie in the second quadrant. This can be seen clearer after changing the display. Select

MODEL ↓

Geometry → Zoom → Numerical axes limits

Display axes limits			
Horizontal axes			
Left	<input type="text" value="-20"/>	Right	<input type="text" value="10"/>
Vertical axes			
Bottom	<input type="text" value="0"/>	Top	<input type="text" value="30"/>
<input type="button" value="Accept"/>		<input type="button" value="Dismiss"/>	

To copy and replicate the winding geometry, it is necessary to define two extra reference points, 0.002 and 0.003. Select

MODEL ↓

Constraints → By keyboard → Length

First constrained point =	<input type="text" value="0.001"/>
Second constrained point =	<input type="text" value="0.002"/>
Length =	<input type="text" value="0sRad"/>
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

MODEL ↓

Constraints → By keyboard → Angle

First constrained point =	<input type="text" value="0.001"/>
Second constrained point =	<input type="text" value="0.002"/>
Angle =	<input type="text" value="S_Alpha*2"/>
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

MODEL ↓

Constraints → By keyboard → Length

First constrained point =	0.001
Second constrained point =	0.003
Length =	0sRad
<div>Accept</div> <div>Dismiss</div>	

MODEL ↓

Constraints → By keyboard → Angle

First constrained point =	0.001
Second constrained point =	0.003
Angle =	90
<div>Accept</div> <div>Dismiss</div>	

A copy of the winding polygon is needed to represent the inactive coil sections of the model. These regions will be modelled as air, but must be included. Select

MODEL ↓

Geometry → Copy polygon → By keyboard

Copy polygon	
Polygon to be copied	<input type="text" value="2"/>
Transformation options	
<input type="checkbox"/> Rotation	
Rotation centre point	<input type="text"/>
Angle of rotation	<input type="text"/>
<input type="checkbox"/> Translation	
X displacement	<input type="text"/>
Y displacement	<input type="text"/>
<input checked="" type="checkbox"/> Reflection	
Reflection plane start point	<input type="text" value="0.001"/>
Reflection plane end point	<input type="text" value="1.03"/>
<div>Accept</div> <div>Quit</div>	

Replications of these polygons can be used to model the entire winding set. The replication for polygon 2, which represents part of the active windings, will be a reflection into the second quadrant.

MODEL ↓

Geometry → Replicate polygon → By keyboard

Polygon number = 2	
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Set the replication parameters	
No. copies x-direction	0
X-dir. copies displacement	0
No. copies y-direction	0
Y-dir. copies displacement	0
<input checked="" type="checkbox"/> reflect on	<input type="checkbox"/> reflect off
Start line reflect at	0.001
End line reflect at	0.003
No. rotation copies	0
Centre pt of rotation	
Angle of rotation	0
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

For polygon 3, the replication must first reflect across to the other side of the pole and then be rotated so that all winding sections are filled.

MODEL ↓

Geometry → Replicate polygon → By keyboard

Polygon number = 3	
<input type="button" value="Accept"/> <input type="button" value="Dismiss"/>	

Set the replication parameters	
No. copies x-direction	<input type="text" value="0"/>
X-dir. copies displacement	<input type="text" value="0"/>
No. copies y-direction	<input type="text" value="0"/>
Y-dir. copies displacement	<input type="text" value="0"/>
<input checked="" type="checkbox"/> reflect on	<input type="checkbox"/> reflect off
Start line reflect at	<input type="text" value="0.001"/>
End line reflect at	<input type="text" value="0.002"/>
No. rotation copies	<input type="text" value="(S_poles/2) - 2"/>
Centre pt of rotation	<input type="text" value="0.001"/>
Angle of rotation	<input type="text" value="S_Alpha*2"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

Material Properties

The material properties of the regions will now be set. The stator will be non-conducting and will normally be run as a non-linear problem, so permeability need not be parameterized. Therefore, set the material properties as

- A non-linear material, labelled **STEEL**
- Zero Conductivity
- A permeability of 1000 (for linear testing purposes)
- A subdivision size based on the outer radius

MODEL ↓

Material data → By keyboard

Polygon number =	<input type="text" value="1"/>
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Set the material properties	
<input checked="" type="checkbox"/> Polygon	<input type="checkbox"/> Background
AIR, CONDUCTOR, IRON	STEEL
Maximum subdivision	(OsRad-IsRad) / 10
Permeability	1000
Conductivity	0
Current density	0
Phase of the region	0
Velocity of region	0
Conductor number	0
<input checked="" type="checkbox"/> Connected	<input type="checkbox"/> Disconnected
Include region in OP2 file	1
Accept	Quit

The material properties for the windings are also to be set. Both replications for the winding have the same current density, and the same sign for the current density. The system variable **AREA** can be used within material properties to calculate the area of the region.

Select

MODEL ↓

Material data → By keyboard

Polygon number =	2
Accept	Dismiss

Set the material properties

☒ Polygon

AIR, CONDUCTOR, IRON

Maximum subdivision

Permeability

Conductivity

Current density

Phase of the region

Velocity of region

Conductor number

☒ Connected

Include region in OP2 file

Accept

☐ Background

CONDUCTOR

$(OsRad - IsRad) / 20$

1

0

$Nturns * Curr / AREA$

0

0

0

☐ Disconnected

1

Quit

Polygon 3 is set to have the magnetic properties of air, as it is representing the inactive drive coils and has no current present. Select

MODEL ↓

Material data → By keyboard

Polygon number = 3

Accept

Dismiss

Set the material properties	
<input checked="" type="checkbox"/> Polygon	<input type="checkbox"/> Background
AIR, CONDUCTOR, IRON	AIR
Maximum subdivision	(OsRad-IsRad) / 10
Permeability	1
Conductivity	0
Current density	0
Phase of the region	0
Velocity of region	0
Conductor number	0
<input checked="" type="checkbox"/> Connected	<input type="checkbox"/> Disconnected
Include region in OP2 file	1
Accept	Quit

As the stator is a ferrous type material, it requires a suitable BH curve to describe its magnetic characteristic properties. In this example, the default curve that is attributed to material **IRON** will be used as the **STEEL** BH curve. Hence select:

MODEL ↓

BH or DE Data → IRON → Store in file

and store the BH data as file name *default.bh*. The BH characteristic of the stator material, **STEEL**, is set by

MODEL ↓

BH or DE Data → STEEL → Load from file

Choose the BH data file *default.bh* from the list and **Accept**.

Improving the mesh

The model will now be checked to ensure that it produces a good model and mesh, although it will not be possible to analyse it as the model is incomplete. Select

MODEL ↓

Check OPERA-2d data

This will convert the model to the OPERA-2d data form. A list of checks are made. Warnings about unmatched symmetry sides are reported, but can be ignored as the periodicity condition used to match symmetry sides will not be applied until later.

The mesh can be viewed using

MODEL ↓

Check OPERA-2d data → Generate mesh

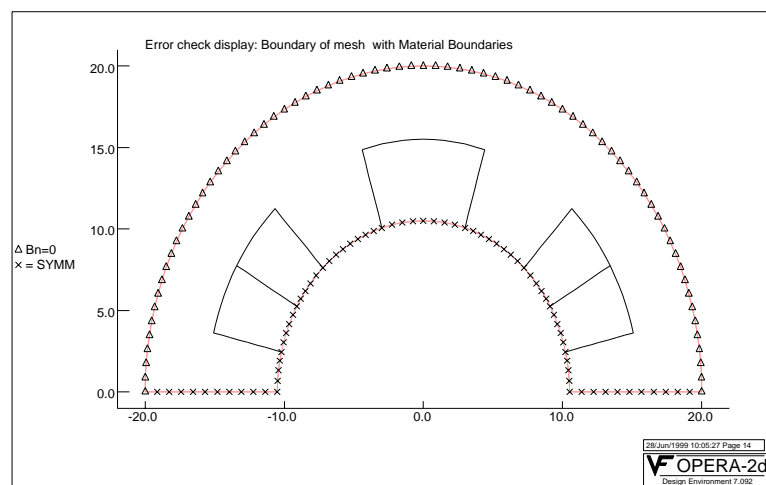


Figure 6.10 Boundary plot generated by the mesh generation

MODEL ↓

Check

OPERA-2d data → View pre-processor model → +mesh (-mesh toggle)

Check

OPERA-2d data → View pre-processor model → Refresh

As with the rotor, it is beneficial to the solution accuracy to improve the mesh at the corners of the stator where flux variation is highest, and the motor is likely to exhibit signs of magnetic saturation. The mesh can be improved using mesh control points. These points increase the mesh density in a local area of the model.

Return to the main MODEL menu and select

MODEL ↓

Geometry → Mesh control point

Set mesh	
Mesh point name	<input type="text" value="1.06"/>
Subdivision at point	<input type="text" value="airgap/3"/>
Radius of influence	<input type="text" value="airgap/3"/>
Associated polygon	<input type="text" value="0"/>
Mesh point options	
<input type="checkbox"/> Delete	<input type="checkbox"/> List <input checked="" type="checkbox"/> None
First point	<input type="text" value="1"/>
Last point	<input type="text" value="*"/>
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

This mesh control point will increase the mesh density near the corner of the stator pole by decreasing the subdivision size. The effect of this mesh control point can be seen by modifying the value of the variable **AIRGAP** to a new value of 0.05 and using the menus described earlier to view the mesh.

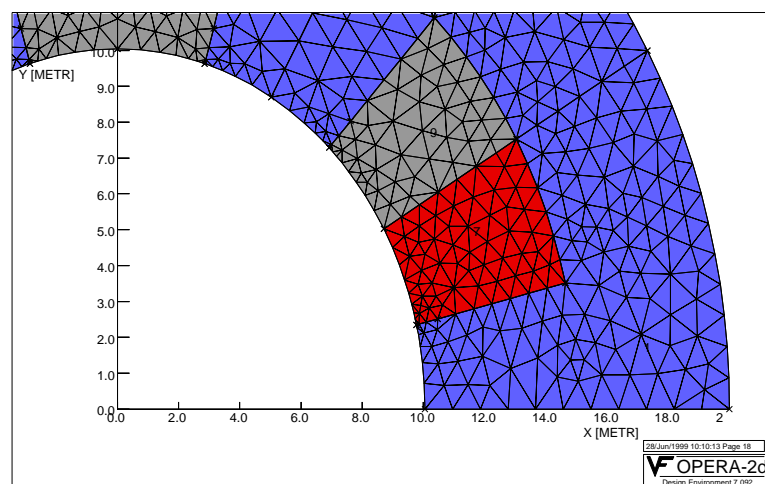


Figure 6.11 Improved mesh density with an airgap of 0.05

Preparing the Stator PART for the User System

The display should be set to the view that the *User System* will display when variables for the Stator are being set. The size of the screen will be stored as expressions so that changes in dimension can be reflected in the part view.

To set the view that the part shows select:

DISPLAY ↓
Hide labels (*Label polygons toggle*)

DISPLAY ↓
Options

Refresh the viewing screen

Display border

X minimum -OsRad

X maximum OsRad/2

Y minimum 0

Y maximum OsRad*1.5

Show nodes:

☒ NONE ☐ ALL ☐ REFE ☐ POLY

Show variable names:

☒ NONE ☐ ALL ☐ SAME

+ or -

Show polygon:

☒ NONE ☐ ALL ☐ SAME

+ or - +1+2

Accept

Quit

DISPLAY ↓
Refresh

The view now only shows the stator and winding polygons. This view is stored and the part created by

DEM ↓

Create or modify a part → Create new part

Part name =	Stator
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The system is now waiting for different items that should be associated with this part. To set a descriptive text message at the top of the user part menu, select:

Add part command → Set text

Text message =	Stator dimensions
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Having set some text, add the variables whose values can be set in this part.

Select **Ask for variable**. and from the list of variables select **S_Poles** followed by **Accept**.

Repeat this for the following 5 variables:

Ask for variable → Osrاد

Ask for variable → S1

Ask for variable → S2

Ask for variable → Coild

The winding properties will also be set within this part. Add a new message and the two variables, **NTurns** and **Curr** Select

Set text

Text message =	Winding properties
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Ask for variable → **NTurns**Ask for variable → **Curr**

The material **STEEL** will need to have a suitable BH curve assigned to it. The user can view the BH file associated with the material in the part menu. Select **Set text**

Complete the dialogue box with the text string and **Accept**

Text message =	Stator BH characteristics
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Select **Show material BH data**

Material name	STEEL
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The **PART** menu created can be demonstrated and tested by selecting

DEM ↓

Demonstrate a part → Stator

This menu is the means by which the key design variables for the rotor can be easily set. The parametric model can be updated with new variable values by modifying values and selecting the **Update** button.

To close the menu select **Quit**.

The stator DEM has now been completed and needs to be saved. From the menu select

DEM ↓

Save as new DEM file

Save the DEM file as *stator.dem*.

Having saved the DEM file, it is necessary to reset the *Configuration System* before starting the stator DEM. Select

OPTIONS ↓

Clear and reset

Motor

DEM Summary

The data which will be defined in this base DEM is listed as follows:

- Solution parameters
- Unit dimension settings
- Finite element analysis settings
- User variables required for the post processing
- Constraints for reference points which are required by the post processing
- The automatic post processing commands needed for analysis
- Symmetry transformations to pair symmetry boundaries

Solution and unit settings

Firstly the solution parameters and units must be set to the appropriate values. This is done by:

MODEL ↓
 Solution settings

Complete the dialogue box as follows and then **Accept**:

Solution Parameters

Solution Symmetry:

☒ XY Symmetry

☐ AXI Symmetry

Solution potential type:

☐ Scalar pot V

☒ Vector pot A

☐ Modified RA (AXI)

Element type:

☒ Linear 3-nodes

☐ Quadratic 6-nodes

Field type:

☐ Electric

☒ Magnetic

Accept
Quit

Select **Return** to close the solution menu options. The model units must be set.

UNITS ↓

Length unit → Centimetre

and **Return**.

UNITS ↓

Density unit → Amps/cm**2

and **Return**.

The Finite Element Analysis Parameters

The Design Environment prepares files directly for analysis by the finite element solvers. The SRM model will be solved by the statics analysis module (the ST solver) with non-linear materials and up to 2 adaptive iterations to help improve the mesh and solution accuracy. The data necessary to run the ST solver is defined by

MODEL ↓

Analysis data → Static analysis (ST) → Non-linear analysis

Non-linear iteration data	
Iteration type	<input type="checkbox"/> Simple <input checked="" type="checkbox"/> Newton
Number of iterations	<input type="text" value="21"/>
Tolerance	<input type="text" value="0.001"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

MODEL ↓

Analysis data → Static analysis (ST) → Mesh refinement options

Maximum number of iterations =	2
Maximum number of elements =	*
Final convergence accuracy(%) =	1
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Design Variables

The key variables used in the post processing need to be defined. This is done using the **VARIABLE** command. In each case the variable name will be given, followed by a value to which the variable is to be set and a description of the variable for later use in the *User System*.

The first variable to be set is **Orrad**.

MODEL ↓

Variables → Define new variable

Variable name =	Orrad
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

Expression =	10
Descriptor =	Outer radius
<input type="button" value="Accept"/>	<input type="button" value="Dismiss"/>

The menu route used above should be used to configure the remaining variables used in base DEM. The variables to be defined are listed below:

Variable name	Expression	Descriptor
AirGap	0.5	Airgap length
Rangle	0	Rotation angle

The expression given for the **Airgap** is probably too large, but this expression can be easily changed to a more reasonable value at a later stage.

The Automatic Post Processing

The aim of this post processing is to establish a value of torque that the SRM will generate for various geometric configurations and rotor angles. This will be achieved through a maxwell stress calculation using a line integral in the airgap between rotor and stator.

A reference point is needed to mark the centre of the circular arc used in the integration. This point is created and constrained by

MODEL ↓

Constraints → By keyboard → Fixed point

Point name =	0.001
X position =	0
Y position =	0
<div>Accept</div> <div>Dismiss</div>	

This generates a new reference point 0.001 and places it at the origin.

The line integral command can be defined to perform a line integral through the centre of the airgap through an arc of 180 degrees (starting at the rotation angle of the rotor). Add the line integral post processing command by

DEM ↓

Post processing → Add command → INTC

Circular Line integral (INTC)			
Centre point	<input type="text" value="0.001"/>	Arc radius	<input type="text" value="OrRad+AirGap/2"/>
Start angle	<input type="text" value="RAngle"/>	End angle	<input type="text" value="RAngle+180"/>
No. of points	<input type="text" value="10000"/>	Time	<input type="text" value="0"/>
Component	<input type="text" value="Pot"/>		
Averaged fields:	<input checked="" type="checkbox"/> Yes <input type="checkbox"/> No		
Torque action	<input type="text" value="0.001"/>		
Accumulator option:			
<input checked="" type="checkbox"/> Zero <input type="checkbox"/> Add <input type="checkbox"/> Subtract			
<input type="button" value="Accept"/>		<input type="button" value="Dismiss"/>	

A message box will appear showing the **INTC** command entered with its parameters. The message will also indicate if any errors are present within the command. The value of torque needs to be doubled to account for the symmetry and stored in a temporary variable. Select

DEM ↓

Post processing → Add command → \$CONSTANT

\$CONS command	
Variable name	<input type="text" value="#T"/>
Set value	<input type="text" value="TORQUE*2"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

The resultant torque of the sum of these two integrations will be written, along with the rotor angle, to a results file.

DEM ↓

Post processing → Add command → RESULTS

RESULTS command	
Description	Rotor angle
Value	Rangle
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

DEM ↓

Post processing → Add command → RESULTS

RESULTS command	
Description	Output Torque
Value	#T
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

The post processing commands can be viewed as they will appear in the post processing command file by selecting

DEM ↓

Post processing → Show command file

Setting the symmetry transformation

As half a model of the SRM is used, a symmetry transformation is required to match the sides of the rotor, stator and airgap polygons, which have the symmetry boundary condition. This can be achieved by

MODEL ↓

Periodic symmetry

Set symmetry transformations:	
Symmetry options:	
<input type="checkbox"/> POSItive	<input checked="" type="checkbox"/> NEGAtive
<input type="checkbox"/> LIST	<input type="checkbox"/> DELEte
Rotation centre point	<input type="text" value="0.001"/>
Rotation angle	<input type="text" value="180"/>
X-shift transformations	<input type="text" value="0"/>
Y-shift transformations	<input type="text" value="0"/>
<input type="button" value="Accept"/>	<input type="button" value="Quit"/>

to apply rotate symmetry faces by 180 degrees about point 0.001 to match them with the periodic neighbour.

The base module for the motor has now been completed and needs to be saved. Select

FILE ↓

Save as new DEM file

Save the DEM file as *base.dem*.

Having saved the DEM file, it is necessary to reset the *Configuration System* before starting the rotor DEM. This is to remove the variables, constraints and reference points. Hence select:

OPTIONS ↓

Clear and Reset

Model testing

All the parametric data necessary to generate a SRM model has been completed and exists in 3 independent DEM files. These files need to be brought together to complete the model. The data will also need to be tested for any errors or inconsistencies.

The base DEM file is the first file to be loaded in. Hence select

FILE ↓
 Load DEM file

select *base.dem* from the file selection box and **Accept**.

The two remaining DEMs will be appended to this file. Select

FILE ↓
 Append DEM file

and select *rotor.dem*. The parametric model of the rotor will be appended to the data already resident.

Select

FILE ↓
 Append DEM file

and append *stator.dem*. A complete model of a SRM will now exist.

When a DEM file is appended, variables which are present in both files take their value from the initial file and not from the file being appended. Therefore their values or expressions are not updated.

The data should be tested to locate any errors needing correction. The command which is used to generate the pre processing files can also be used to test the integrity of the data. This is necessary in checking for the existence of errors and also inspecting the model FE mesh. The **CONVERT** command is used for this by selecting

MODEL ↓
 Check OPERA-2d data

The OPERA-2d pre and post processor **CHECK** command is automatically run. Data messages are directed to the screen and contain the errors and warnings associated with the conversion process. There should be zero errors for successful data conversion and ideally zero warnings.

Within the **CONVERT** command are several sub-commands. These commands are taken from the OPERA-2d pre and post processor and allow the finite element mesh generation (**MESH** command), display of the OPERA-2d model (**RECO** command), printing of OPERA-2d model data (**PRINT** command) and writing of the OPERA-2d data file for analysis (**WRIT** command). These commands should be used to test the validity of the models generated for a range of variable values, e.g.

- with the rotor angle of non-zero, check that the symmetry conditions have been correctly applied to all sides
- Check that the models are built correctly when the airgap is at the minimum allowed value

Once satisfied that there are no errors or inconsistencies in the model, save the model as the single module file for use within the User System (see “**Switched Reluctance Motor User System**” on page 5-1) by

FILE ↓
 Save as new DEM file

and save as *srm.dem*.

INDEX

Symbols

\$ commands 4-30
\$ CONS 6-56

-A-

analysis 3-8, 5-5
analysis settings 6-53
AREA system variable 4-17
automatic analysis 3-8
axisymmetric 3-2

-B-

BH data 4-23
boundary conditions 4-27
 potential 4-27

-C-

conductivity 4-3
constraint
 angle 6-35, 6-36, 6-40, 6-41
 fixed point 4-7, 4-20, 4-36
 length 6-7, 6-40, 6-41
 length difference 6-8, 6-12
 vector 4-7, 4-9, 4-25, 4-40
current density 4-3, 4-17
curvature 6-16

-D-

dimension
 line style 4-38

options 4-37
display options 4-33, 4-36, 4-39, 4-43, 6-49

-F-

files

control 3-3, 5-3
dem 4-31, 4-44
loading 3-2, 5-1
model variations 3-4, 3-9, 5-3
post processing command scripts 6-57
results 3-9, 5-6
testing OPERA-2d data 6-59

-I-

INTA command 4-30
INTC 6-56

-L-

limits 4-10, 4-16, 4-18, 4-21
link 6-11
list
 constraints 4-10
 limits 4-11
 polygon data 4-8
 polygons 4-9
 variables 4-6
listing variables 6-6

-M-

material properties .. 4-12, 4-17, 4-26, 6-18,
6-19, 6-46

mesh

- subdivisions 6-18

model variations 3-4, 3-9

- generating 3-6
- loading 5-6
- storing 3-6, 5-5

modify

- constraint 4-8

-N-

non-linear materials 4-23

-P-**PART**

- command 4-34
- creating 4-32, 6-25
- list commands 4-35

PART command 4-44

parts 4-32

permeability 4-22

points

- reference 4-35

polygons

- creating 6-6
- subdivisions 6-18

post processing 3-8, 4-30, 5-5, 6-56

power loss 4-30

-R-

range variables 3-6, 5-5

reference points 4-35

reflections 6-19, 6-46

replications 6-19, 6-46

RESULTS 5-6, 6-56

rotations 6-19, 6-46

running analysis 3-3, 5-3

RUNPART command 3-4, 5-3

-S-

save dem file 4-44

sides

- curvature 6-16
- links 6-11

solution parameters

- element 6-53
- field 6-53
- potential 6-53
- symmetry 6-53

SOLVE command 6-53

SRM 5-1

storing model variations 5-5

subdivisions 4-12, 4-17, 4-22, 6-18

switched reluctance motor 5-1

-T-

testing model data 6-59

-U-

units 4-3, 6-53

User System 5-1, 6-24

- settings 4-32

-V-

variables

- descriptions 6-4
- expressions 6-4
- listing 6-6
- range 3-4, 5-5
- setting 3-4

-W-

WRITE command 4-31